ERDC/CHL TR-00-16

Coastal and Hydraulics Laboratory

US Army Corps of Engineers®
Engineer Research and Development Center

Riverflow/River Stage Prediction for Military Applications Using Artificial Neural Network Modeling

Bernard B. Hsieh and Charles L. Bartos

August 2000

ERDC/CHL TR-00-16
August 2000

Riverflow/River Stage Prediction for Military Applications Using Artificial Neural Network Modeling

by Bernard B. Hsieh, Charles L. Bartos

Coastal and Hydraulics Laboratory
U.S. Army Engineer Research and Development Center
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Final report

Prepared for Military Hydrology Program
U.S. Army Engineer Research and Development Center
Vicksburg, MS 39180-6199

Engineer Research and Development Center Cataloging-in-Publication Data

Hsieh, Bernard Bor-Nian.
Riverflow/river stage prediction for military applications using artificial neural network modeling / by Bernard B. Hsieh, Charles L. Bartos ; prepared for Military Hydrology Program, U.S. Army Engineer Research and Development Center.
92 p. : ill. ; 28 cm. -- (ERDC/CHL ; TR-00-16)
Includes bibliographic references.
1. Streamflow -- Computer programs. 2. Stream measurements -- Computer programs. 3. Neural networks (Computer science) 4. Artificial intelligence. I. Bartos, Charles L. II. United States. Army. Corps of Engineers. III. U.S. Army Engineer Research and Development Center. IV. Coastal and Hydraulics Laboratory (U.S.) V. Military Hydrology Program (U.S.) VI. Title. VII. Series: ERDC/CHL TR ; 00-16.
TA7 E8 no.ERDC/CHL TR-00-16

Contents

List of Figures

List of Tables

Preface

A modern computational tool, artificial neural networks (ANNs), was used to construct a series of prediction and forecasting models for two different scale watersheds – the Sava River and a segment of the Mississippi River. The purpose of this study is to provide another useful tool for military operation. The Military Hydrology Program, Coastal and Hydraulics Laboratory (CHL), Vicksburg, MS, U.S. Army Engineer Research and Development Center (ERDC), funded this study.

This study was conducted in CHL during the period January 1999 to September 1999 under the direction of Dr. James R. Houston, Director, CHL; Mr. Thomas W. Richardson, Assistant Director, CHL; Dr. William H. McAnally, Chief, Estuaries and Hydro-Sciences Division, CHL; and Mr. Thomas J. Pokrefke, Jr., Chief, Modeling Systems Branch, Estuaries and Hydro-Sciences Division.

The study was conducted by Dr. Bernard B. Hsieh and CPT Charles L. Bartos, Modeling Systems Branch. Dr. William D. Martin and Mr. Thomas L. Engdahl, Watershed Systems Group, Modeling Systems Branch, provided much guidance and information for the Sava River watershed. Drs. Kuo-Lin Hsu and Hoshin V. Gupta, University of Arizona, Tucson, AZ, provided technical peer review for the report. Drs. Bin Zhang and R. S. Govindaraju, Purdue University, West Lafayette, IN, provided technical assistance.

During publication of this report, Dr. James R. Houston was Director of ERDC, and COL James S. Weller, EN, was Commander.

1 Introduction

Background

The ability to forecast a river's flow and stage characteristics can be useful in providing a warning to those in the immediate area of impending catastrophic events typically associated with flood conditions. Furthermore, the ability to perform expedient forecasting can assist water resources management personnel in regulating reservoir outflows during low river flows. For military applications, the accurate forecasting of river stage and flow is critical during any military operation since this directly impacts a military unit's force mobility capability.

Most hydrologic processes exhibit a high degree of temporal and spatial variability, and are further plagued by issues of nonlinearity of physical processes, conflicting spatial and temporal scales, and uncertainty in parameter estimates. The capability exists to extract the relationship between the inputs and outputs of such a process, without the physics being explicitly provided. It is also possible to provide a map from one multivariable space to another, given a set of data representing that mapping. These properties of Artificial Neural Networks (ANNs) may be well suited to the problems of estimation and prediction in hydrology.

Two major approaches for modeling the rainfall-runoff or prediction of river flow have been developed in the literature: conceptual (physical basis) modeling and system theoretic modeling. Conceptual models are important in understanding hydrologic processes. There are many practical situations, such as stream flow forecasting, where the main concern is making accurate prediction at specific watershed locations. In a situation for predicting desired locations, a hydrologist may prefer not to expend the time and efforts required in developing and implementing a conceptual model or numerical model, but instead implement a simpler system theoretic model. ANNs provide the capability to supplement hydrological modeling in a fraction of the time.

The development of such system theoretic hydrologic models, both the linear modeling approach and nonlinear decomposition and recursive parameter approach has been used for many years. However, the former approach does not attempt to represent the nonlinear dynamics inherent in the transformation of input series to output series and therefore may not always perform well. The latter approach, the nonlinear decomposition and recursive parameter approach, allows the model parameters to vary with time and can to some extent compensate for the model structure errors that arise from such assumptions. The computational procedures are complicated.

Recently, significant progress in the fields of nonlinear pattern recognition and system control theory have made this possible through advanced computational techniques available through ANNs. The significant contribution of this modern technique is to solve the nonlinearity and time-delay problem for hydrological applications.

Applications of ANNs in rainfall-runoff modeling and stream flow forecasting have been described in many sources. The algorithms to perform these approaches were from backpropagation (Hjelmfelt and Wang 1996), time-delayed (Karunanithi et al. 1994), recurrent C (Carriere, Mobaghegh, and Gaskari, 1996), radial-basis function (Fernando and Jayawardena 1998), modular (Zhang and Govindaraju 1998), to self-organizing (Hsu, Gupta and Sorooshian 1998). It is noted that only one reference for each algorithm is cited.

Study Objective and Scope

The objective: To demonstrate the applicability of the system theoretic ANN approach in developing effective nonlinear models of a river stage, river flow forecasting process without the need to explicitly represent the internal hydrologic structure of a watershed model. The Sava River watershed was of particular interest, with its variable hydrological conditions and its previous hydrologic analysis in support of military operations. A large-scale watershed, such as the Mississippi River, was also analyzed to further demonstrate the capability of flood forecasting (rainfall-runoff process) by ANNs.

In the Sava River analysis, data from the various stream gages was used as input. The goal was to determine and select input from those sites that afforded the maximum warning time for military operations further downstream.

This report provides a general overview of ANNs and important concepts associated with them. (Chapter 2 and Appendix A). This report also addresses the approach used in selecting the best algorithm to operate an ANN for hydrological forecasting (Appendix B) and to discuss the practical implementation of an ANN model (Appendix C) when utilizing commercially available software to perform its operation. Furthermore, this report analyzes what is required to construct several ANN models with different time-scales for two totally different watersheds (Chapter 3 and 4). In follow-on discussion (Chapter 5), the following areas are highlighted: forecasting reliability due to the length of a training record, approach algorithms for use in different situations, data arrangement order for training, cross-validation, testing, and finally, the testing accuracy due to the selection of activation functions, and the performance of testing accuracy due to data representation. As a final wrap-up, technical findings and recommendations are stated in the conclusion.

2 An Introduction of the Computation Tool—Artificial Neural Networks (ANN)

This chapter introduces a new computation tool, Artificial Neural Networks (ANN), used to conduct a hydrological analysis of two watersheds using flow and gage data. The appendixes in this report provide a more detailed description of the ANN modeling methodology used. Appendix A describes the learning strategies, major components for building ANNs, and procedures used to construct an ANN model. Appendix B lists the basic ANN algorithms used for hydrologic forecasting systems with an emphasis on the Back–Propagation (BP) algorithm. Appendix C illustrates the procedures used with the software, NeuroSolutions for Excel, and discusses the practical implementations this software has for ANN modeling and those associated learning processes.

ANNs Features and Applications

ANNs, quite simply, are computational devices or a universal approximator. They can be implemented in the form of a computer chip or simulated on conventional serial computers. ANN is a type of biologically inspired computational model based on the functioning of the human brain. Like humans, ANNs can learn to recognize patterns by repeated exposure to many different examples. The main feature, besides having the ability to learn, is to associate and to be error-tolerant. Unlike conventional problem-solving algorithms, ANNs can be trained to perform a particular task. This is done by presenting the system with a representative set of examples describing the problem, namely in the form of input and output pairing samples.

ANNs will then extrapolate the mapping between input and output data. After training, the ANN can be used to recognize data that are similar to any of the examples shown during the training phase. The ANNs can even recognize incomplete or noisy data – an important feature that is often used for prediction, diagnosis or control purposes. Furthermore, ANNs have the ability to self-organize, therefore enabling segmentation or coarse coding of data.

Over the last few years, ANNs have seen many successful applications of neural computing in commercial, academic, and military applications. Their benefits include pattern recognition, process control, signal processing, and optimization. ANNs are presently being used to solve a variety of problems such as detection, estimation, discrimination, classification, optimization, prediction, interpolation, extrapolation, clustering, or some combination of these problems in many scientific and engineering applications. Typical hydrological applications that ANNs have the capability for is to model rainfall-runoff, stream flow, groundwater management, water quality simulation, and precipitation phenomena.

Basic Structure of ANN

An ANN's basic processing element is the "node", which is similar to a human's neuron structure. The node has one or more input lines and one or more output lines emulating from it. The input and output lines can be connected to other neighboring nodes to form an artificial "neural-like" network. If the neuron has multiple inputs, it sums the signal that it receives through the several input lines. These processing elements can receive input signals and then output information at a particular strength to the input paths of other processing elements through a connection weight that is modified as the system learns. Besides the input and output layers, there is an intermediate layer of neurons in between called the "hidden layer". Figure 1 shows a fully connected feedforward network with one hidden layer and output layer. Noted: This system consists of *i* nodes for input layer, *j* nodes for hidden layer, and *k* (*k=1* in this figure) nodes for output layer.

A neuron collects information from all preceding neurons relative to the flow of the information and propagates its output to the neurons in the following layer. The output of each preceding neuron is modulated by a correspondent weight before affecting the activity of the neuron. This activity is then modified by an activiation function and becomes the final output of the neutron. The signal is then propagated to the neurons of the next layer. This process is demonstrated as Figure 2.

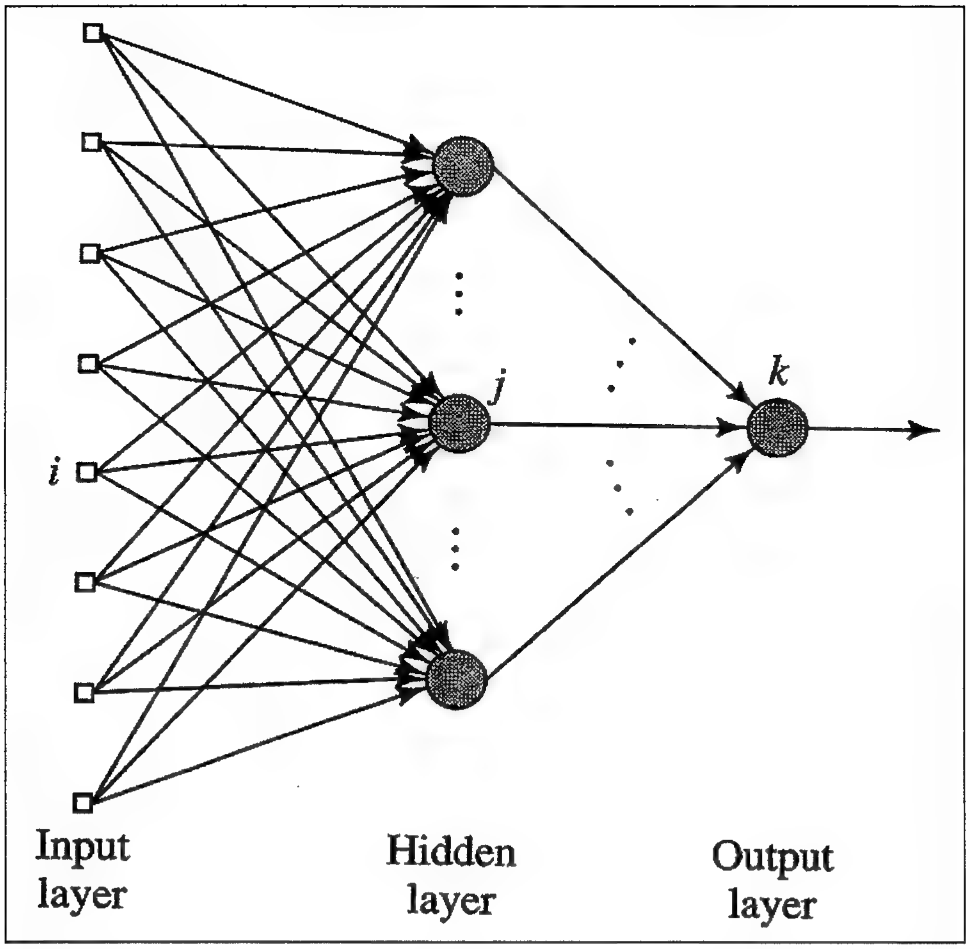


Figure 1. Fully connected feedforward network with one hidden layer and output layer

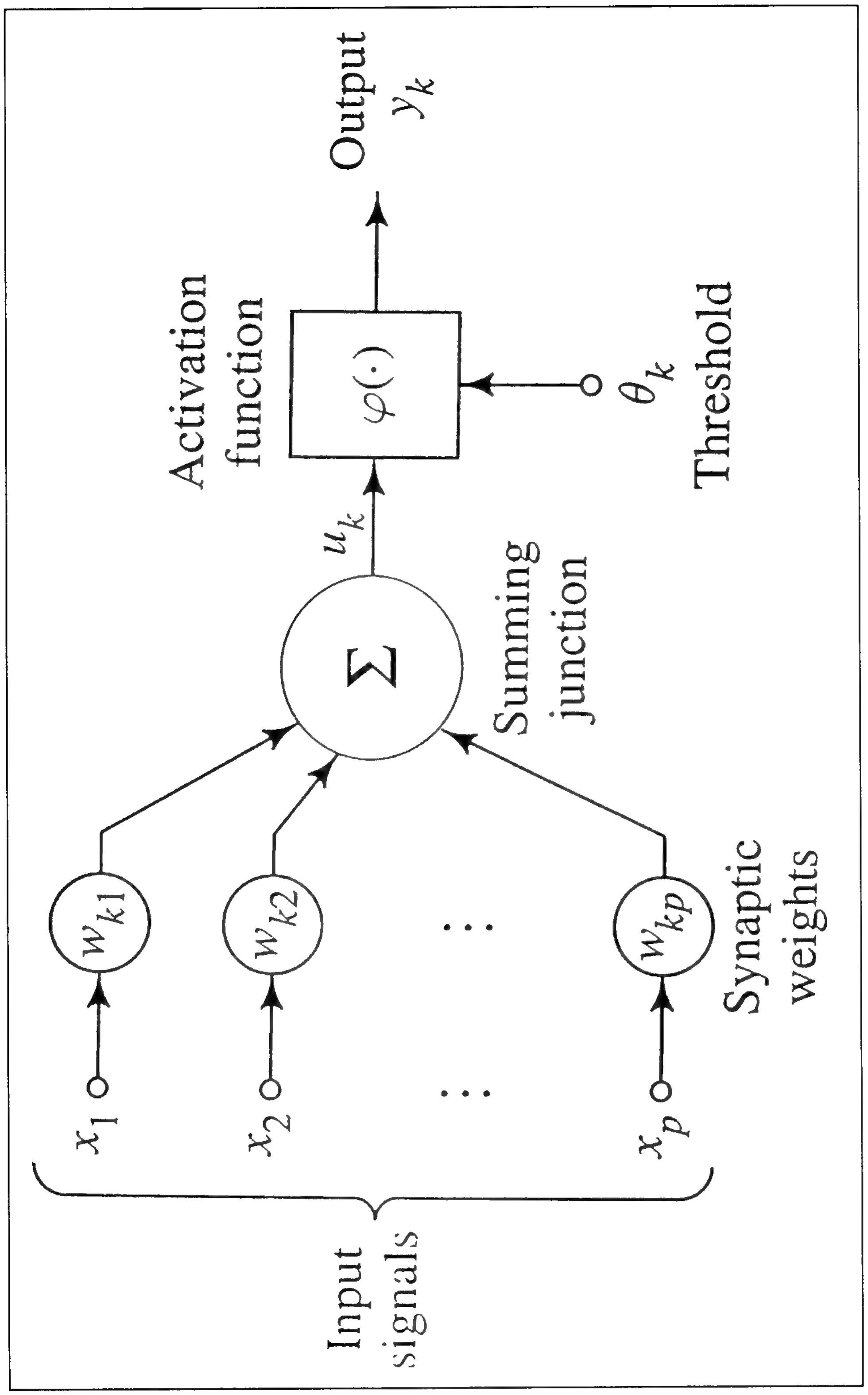


Figure 2. Nonlinear model of a neuron

3 Sava River Application

Sava River watershed and modeling needs for military operations

The Sava River (Figures 3 and 4) is the largest river in the former Yugoslavia. Since Yugoslavia was divided into several new republics, the Sava River's origin is in Slovenia. The Sava River separates the countries of Croatia, Bosnia, and part of Serbia. The total drainage area at the confluence of the Sava and Danube River comprises 96 thousand square kilometers and the watershed length is 2,255 km. The length of the Sava River is 950 km.

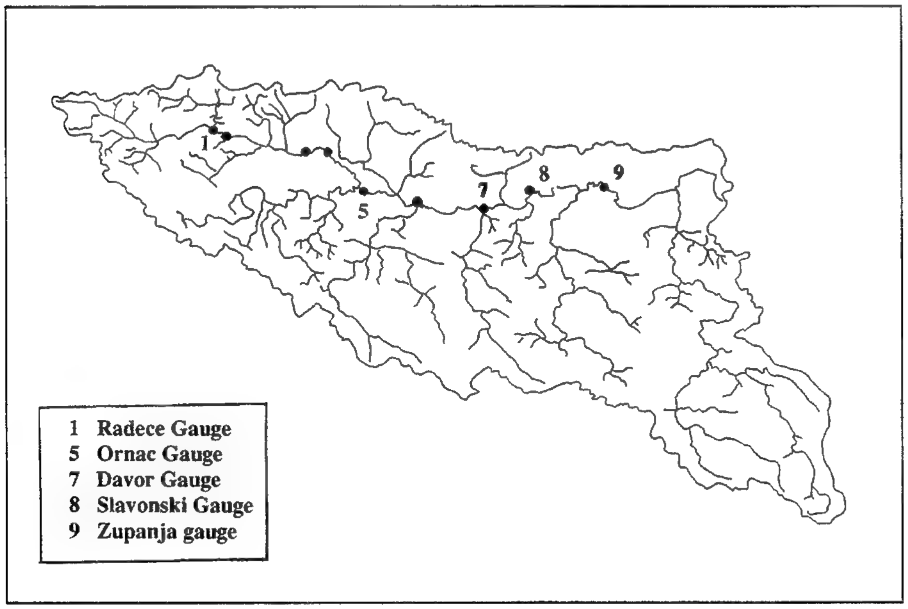


Figure 3. Sava River Basin and its flow stations used for the study

During the peacekeeping mission in Bosnia, the prediction of river stages for military crossing became particularly important. The accuracy of prediction was critical to determine the schedule of military operations, especially the locations where floating bridges were constructed maintained. Therefore, a river flow and stage forecasting system was required to address changes in weather conditions. During the actual operating (Dec 95 - May 97) accurate river forecasts were

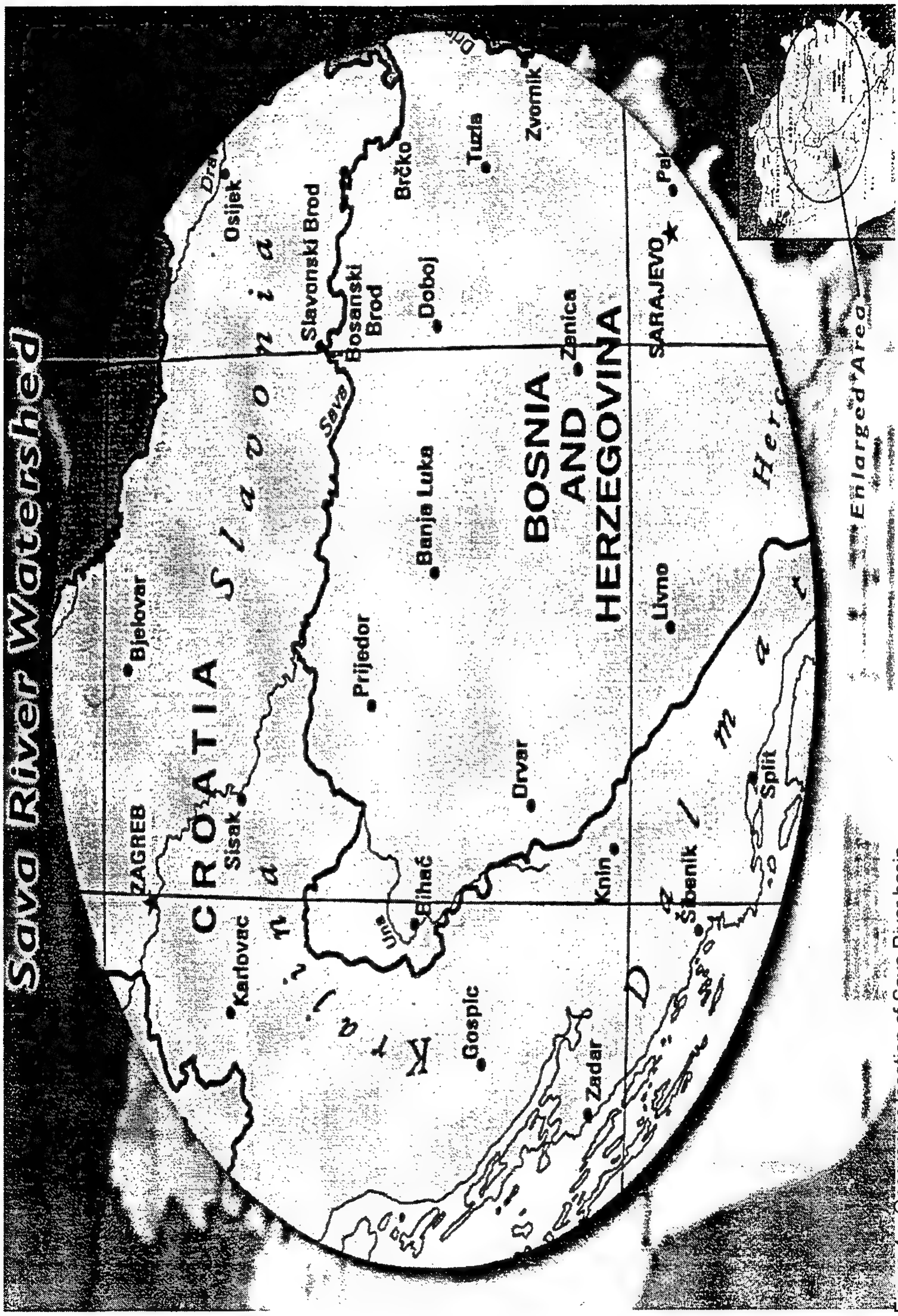


Figure 4. Geographic location of Sava River basin

provided using a combination of river engineering expertise and lumped parameter numerical models. Post fact, the ANN method was applied to determine its ability to provide forecasts for the Sava River.

Database Development for ANN models

A number of river flows and stage data were available from two-dozen gage sites in both the mainstream and tributaries of the Sava River. These were collected during the actual operation. The best data files that can currently be used to construct the model are eight stations for river flow and two stations for river stage. These river flow stations are along the main portion of the Sava River. The data sets include one year of daily mean flow (Figures 5 and 6) and forty years of monthly mean flow (Figures 7 and 8). The two most downstream stations, Zupanja and Slovonski Brod are bridge sites used by NATO forces for military peace keeping operations. In this analysis, these are gage stations (Gages 8 and 9). The daily river stage data (2-1/2 years, Figure 9) is only available for these two most downstream stations. This modeling effort was designed to find an alternative method to predict downstream flow and stage based on the minimum upstream information other than the numerical watershed model simulation.

Sava River Stage Prediction Model

Using the data available, a river stage-forecasting model was constructed using the Slavonski Brod gage (upstream, Gage 8) to predict the Zupanja gage (downstream, Gage 9). The data used consisted of 2.5 years of data. The ANN modeling procedure was divided into one training set (1 year), a cross-validation set (6 months) and a testing set (1 year).

Since this is the first model described in the report, the procedure will be explained in more detail. This model was trained using Multilayer Perceptron (MLP) with a Back Propagation (BP) algorithm having one hidden layer and two input nodes. Using the NeuroSolutions software, the Multilayer Perceptron panel was used to set the parameters that are specific to this neural model. The number of inputs, outputs and exemplars are computed from the input data files. The only parameter to set for the MLP is the number of hidden layers.

The hidden layer panel was used to specify the number of processing elements (PEs), the type of nonlinearity, the type of learning rule, and the learning parameters of the first hidden layer. In this case, the network had two hidden PEs with linear axon type transfer function. The training uses momentum learning with a step size of 1.0 and momentum factor of 0.7.

The output layer panel is the same as the previous panel in that the number of PEs is fixed to the number of output (1). It should be noted that the default step size is one magnitude smaller (set to 0.1) than the previous layer. This is because the error attenuates, as it is back propagated through the network. Since the error

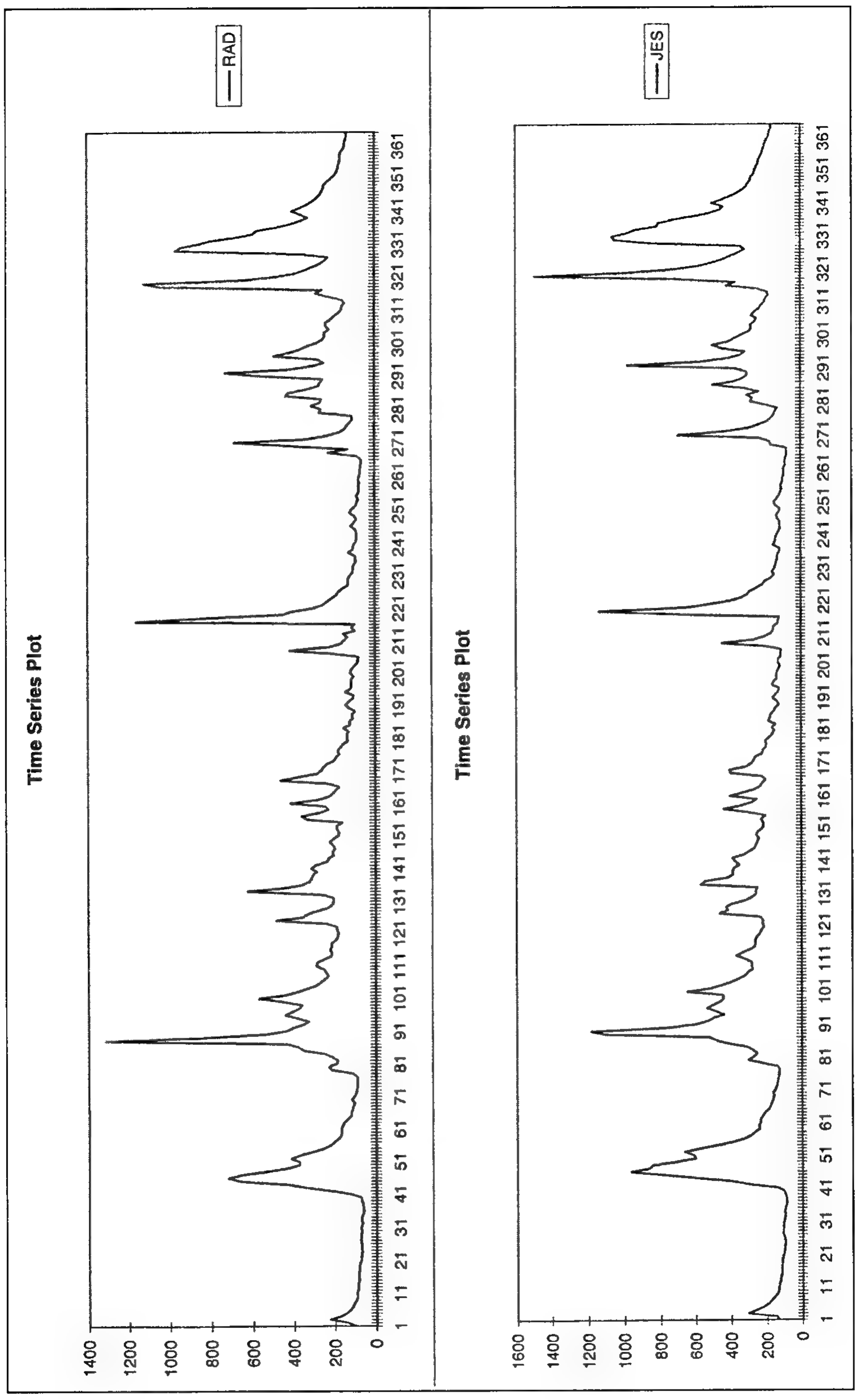


Figure 5a. Daily riverflow (cms) at Radece and Jesenice stations (1987)

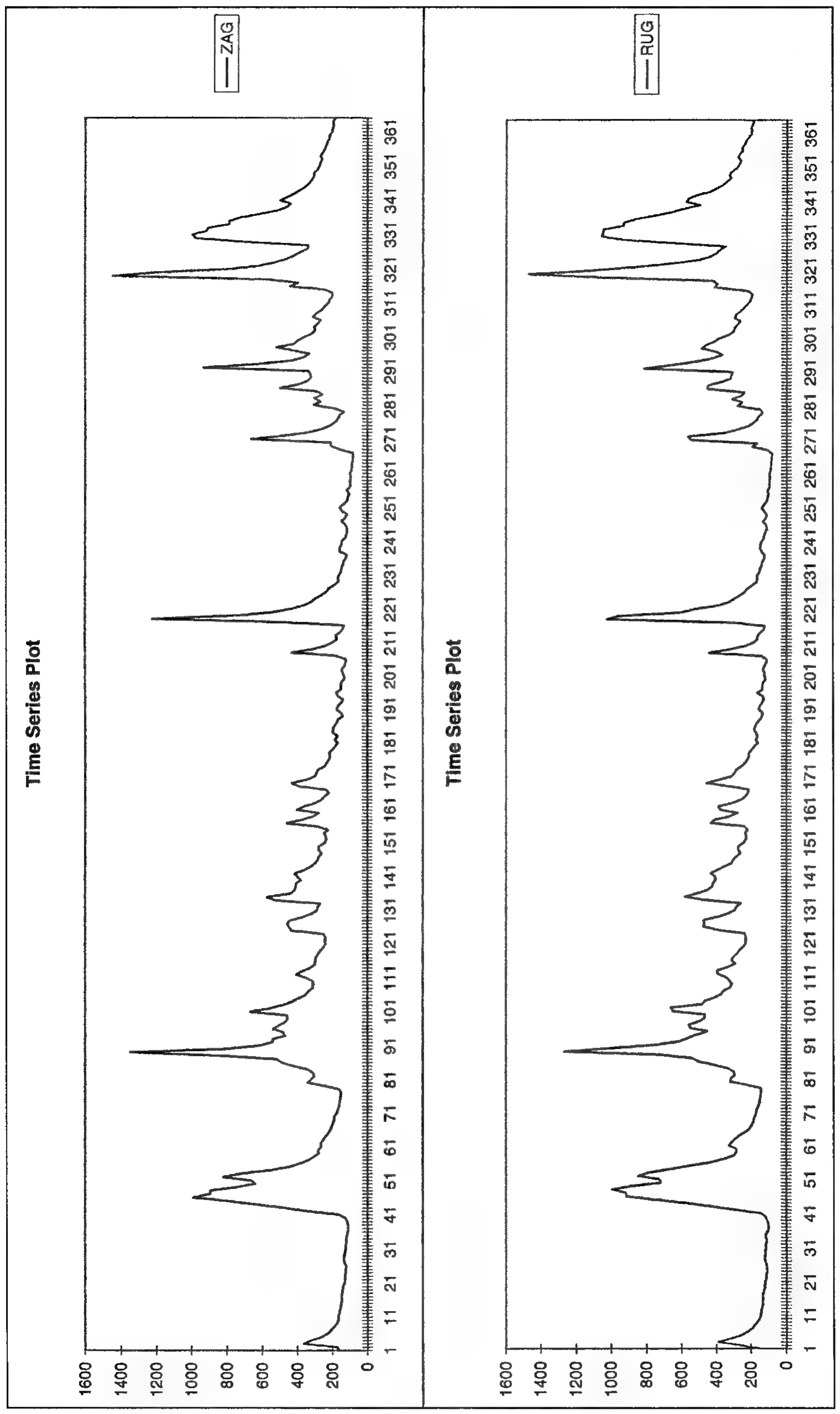


Figure 5b. Daily riverflow (cms) at Zagreb and Rugvica stations (1987)

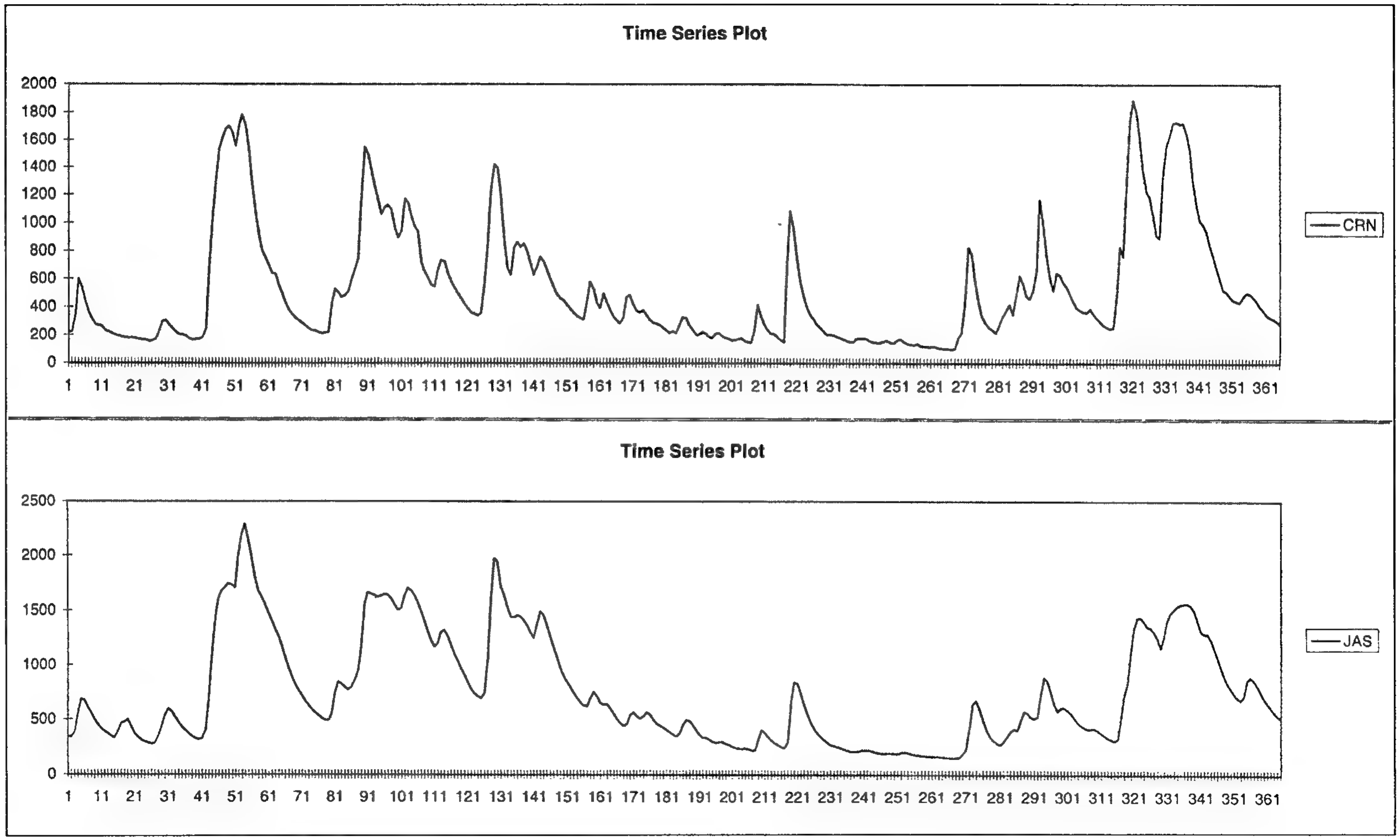


Figure 6a. Daily riverflow (cms) at Crnac and Jasenovac stations (1987)

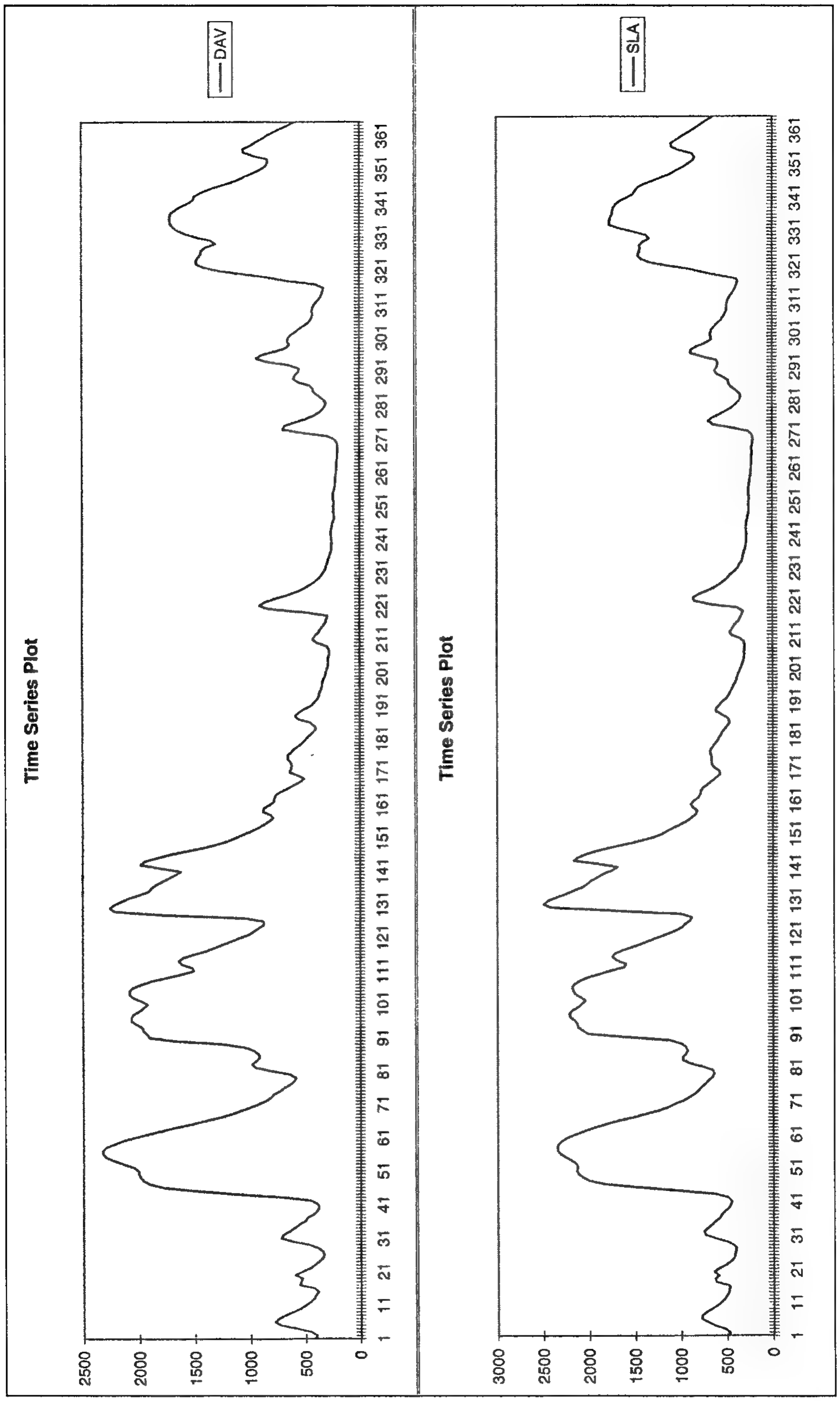


Figure 6b. Daily riverflow (cms) at Davor and Slavonski Brod stations (1987)

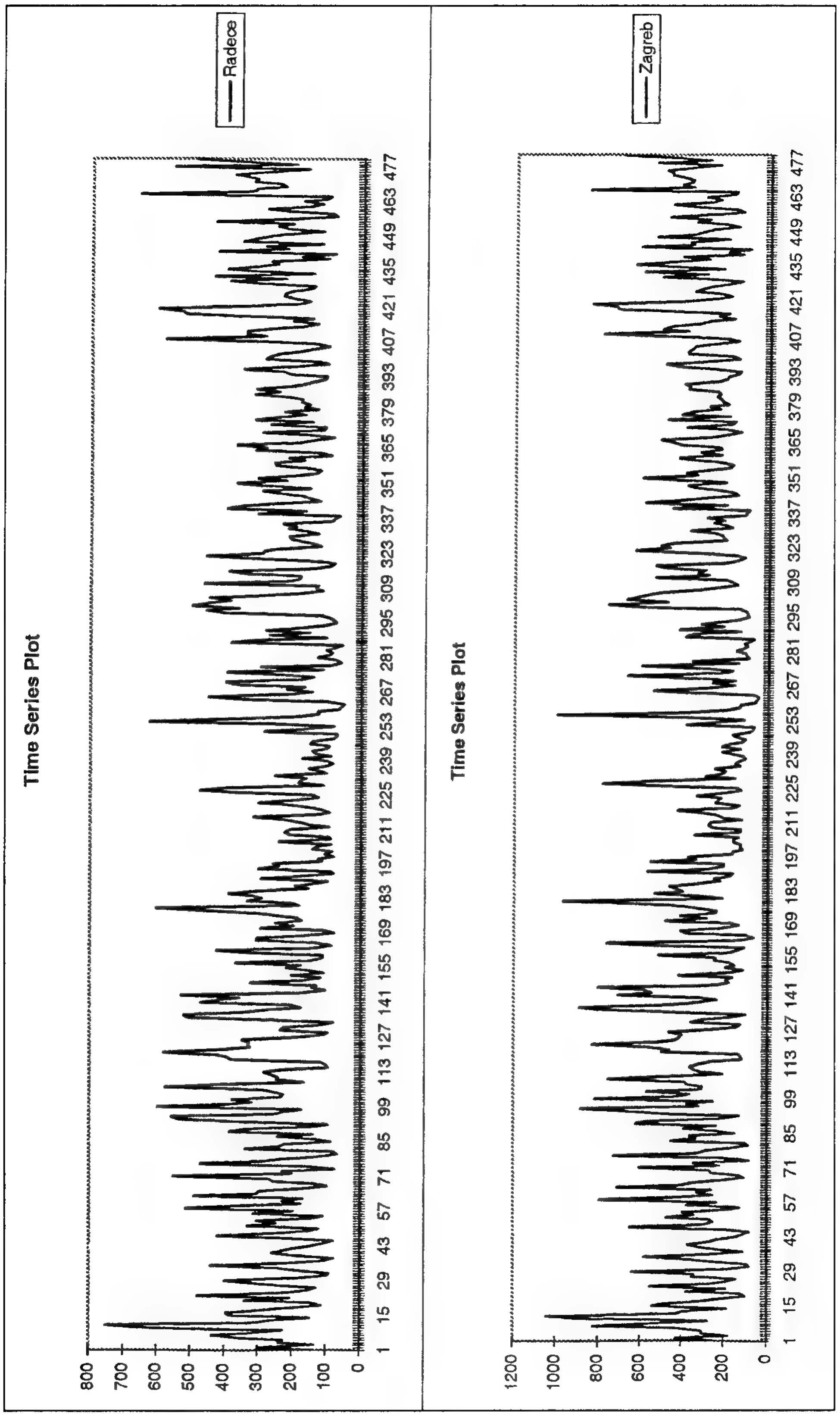


Figure 7a. Monthly mean flow (cms) at Radece and Zagreb stations (1926-1965)

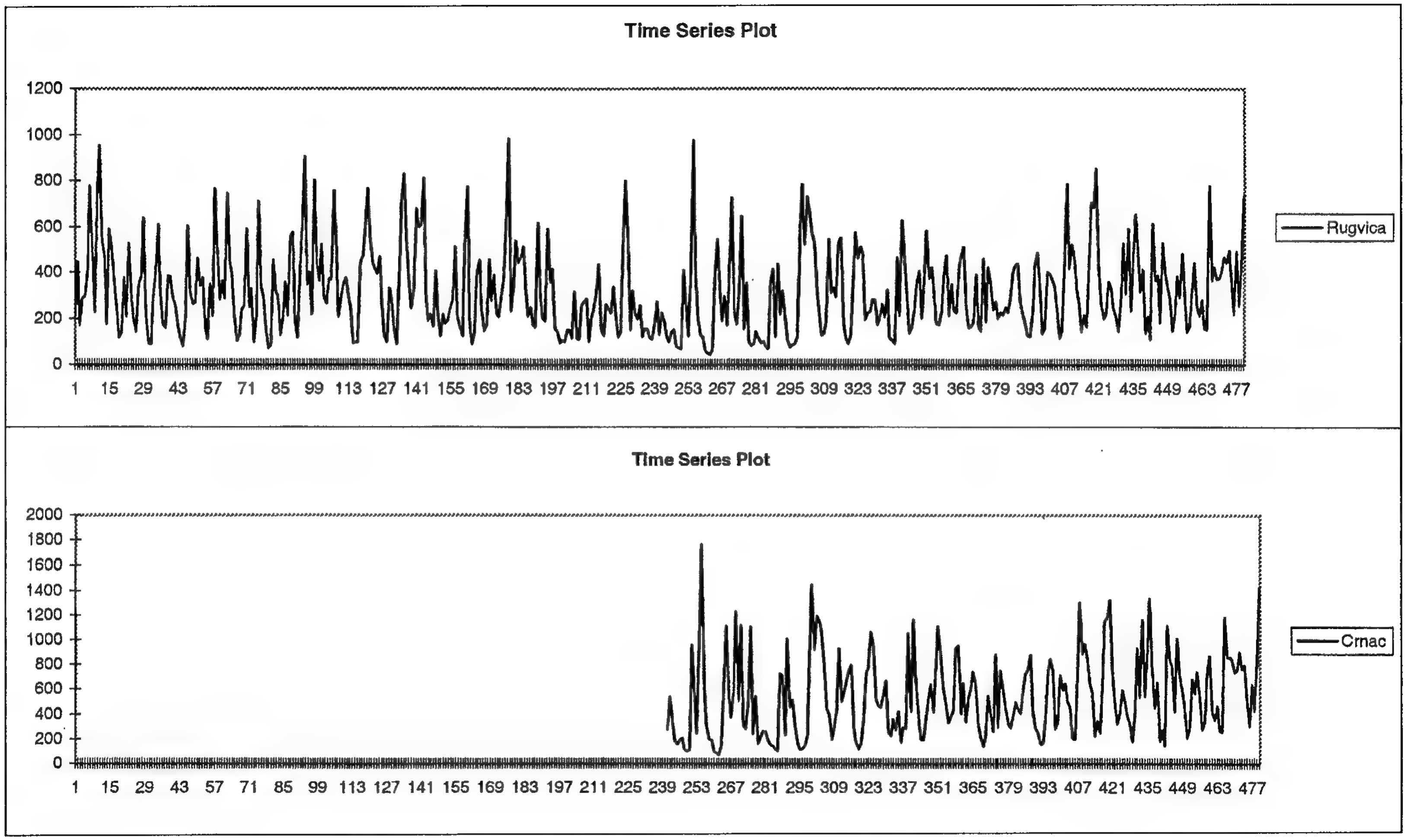


Figure 7b. Monthly mean flow (cms) at Rugvica and Crnac stations (1926-1965)

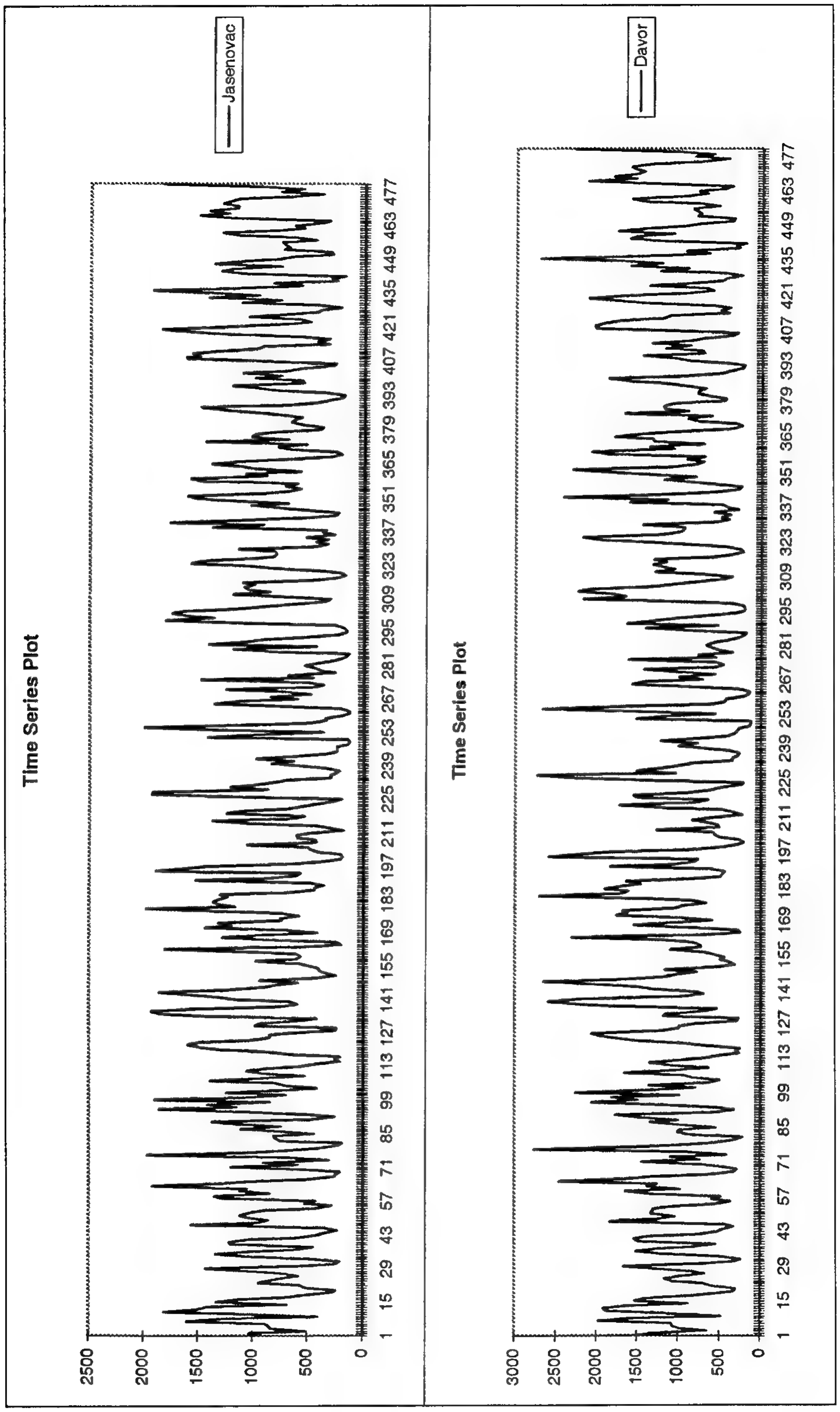


Figure 8a. Monthly mean flow (cms) at Jasenovac and Davor station (1926-1965)

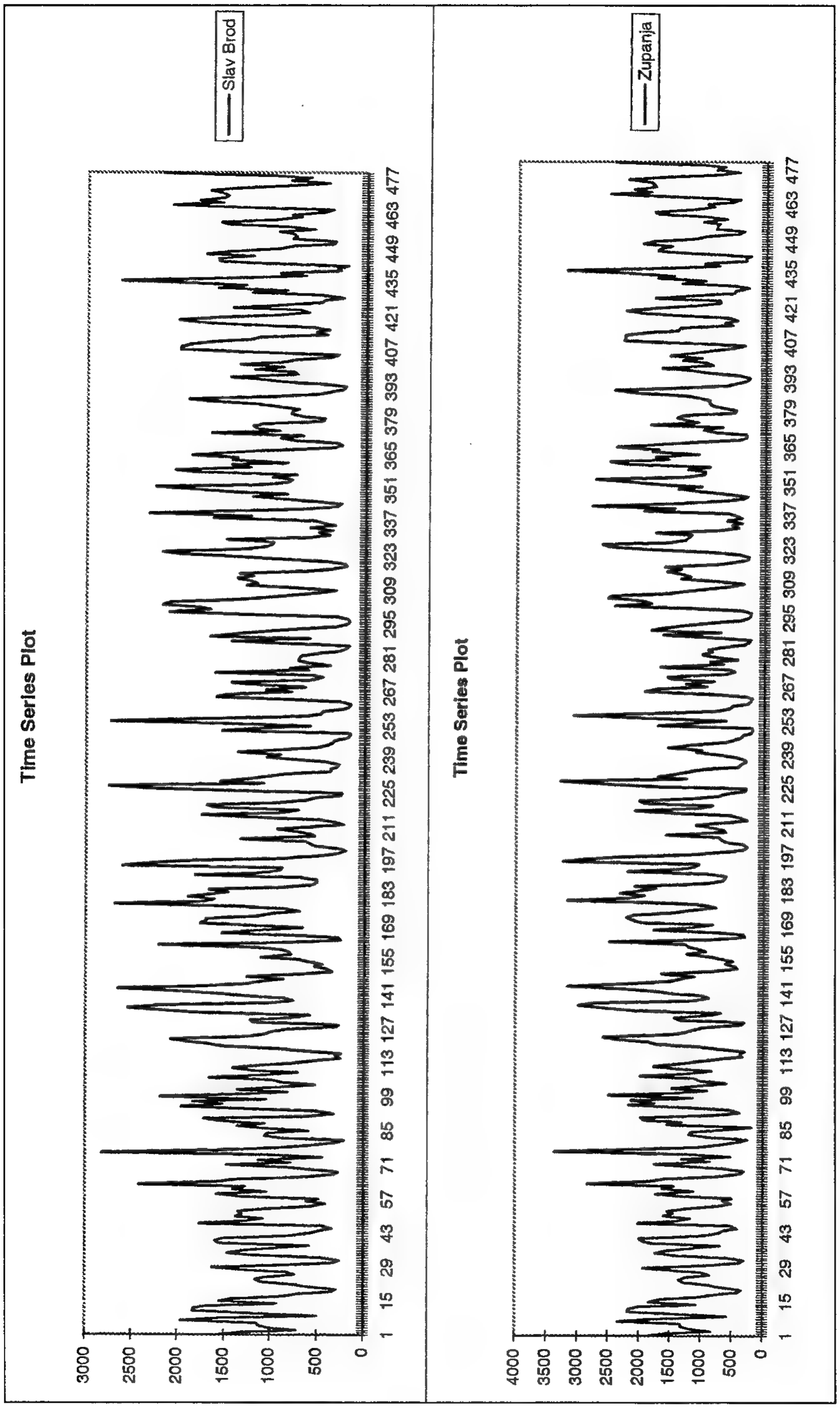


Figure 8b. Monthly mean flow (cms) at Slavoski Brod and Zupanja stations (1926-1965)

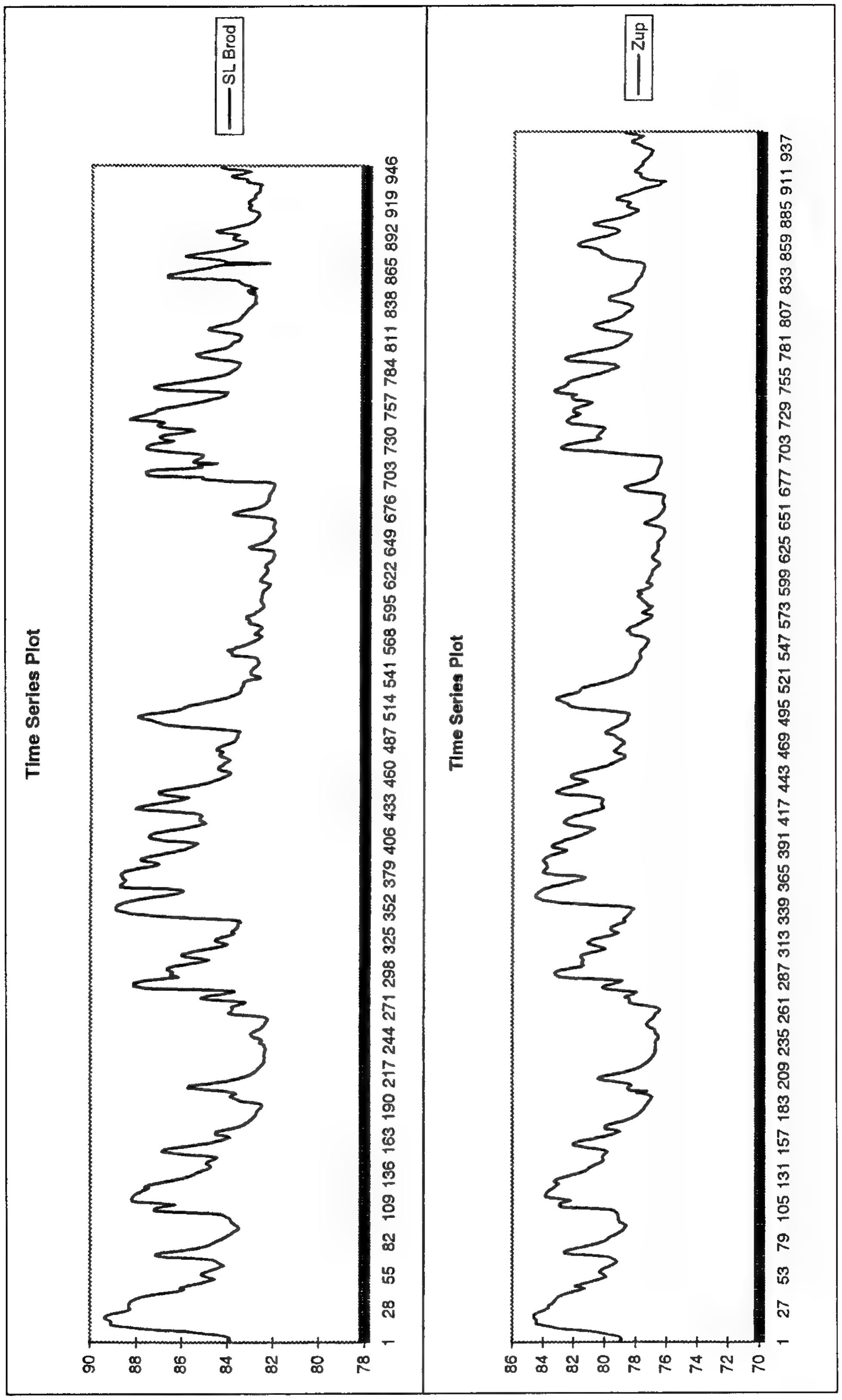


Figure 9a-b. River stages (m) at Slavonski Brod and Zupanja stations (Dec 1995 - July 1998)

is largest towards the output of the network, the output layer requires a smaller step size than that of the hidden layer in order to balance the weight updates.

In the supervised learning panel, the maximum epochs (number of training iterations) are set to 1,000. However, the network may learn the problem in less iterations that this. Usually, the default configuration terminates the training when the error falls below 0.01. Figure 10 summaries the learning curve for the first 50 iterations. It compares the Mean Squared Error (MSE) for both training and cross-validation. At the 20th iteration, the cross-validation has reached its minimum value of the MSE. The optimal weights (total weights is not necessary to be 1) for these stage prediction models are:

Hidden Axon = LinearAxon (0.0636; 0.1251)
Output Axon = LinearAxon (0.1400)
Hidden Weights = (0.7638; -1.7133)
Output Weights =(0.2074;-0. 04714)

Table 1 is a summary of performance based on six statistical measurements. These six measurements are: Mean Squared Error, Normalized Mean Squared Error, Mean Absolute Error, Minimum Absolute Error, Maximum Absolute Error and the Correlation Coefficient. The 3 major components for building an Artificial Neural Network, in this order, are: Training, Cross-Validation and Testing. Based on the results in Table 1, the three stages used in building the ANN show very good training and prediction results with high correlation coefficients and low normalized mean square error. However, with the increasing number of NMSE and the lowering in the numerical value of the correlation coefficient from training to testing, it appears the model is not fully generalized. This is because there is a time lag between these two stations for signal traveling. Figures 11 through 13 show the graphical comparisons between ANN model results and field measurements as training, cross-validation and testing respectively.

In order to examine the model forecasting capability, the forecasting ranges were set up to an analysis period of present to 3 days ahead and several scenarios were conducted. For example, a forecasting model was constructed by using current and previous 2-day stage data at the upstream location (Slavonski Brod) as inputs and using up to 3 days ahead stage data at the downstream location (Zupanja) as the output. This formed a three-input/three-output system. This model was again trained by a MLP with a BP algorithm. The performance analysis of this model is summarized as Table 2. This approach eliminates the loss information for forecasting due to past information. With these results, it is expected about 0.52 meters mean absolute error and a correlation coefficient of 0.911. It also explained the fact that the forecasting reliability decreases with the length of forecasting increases.

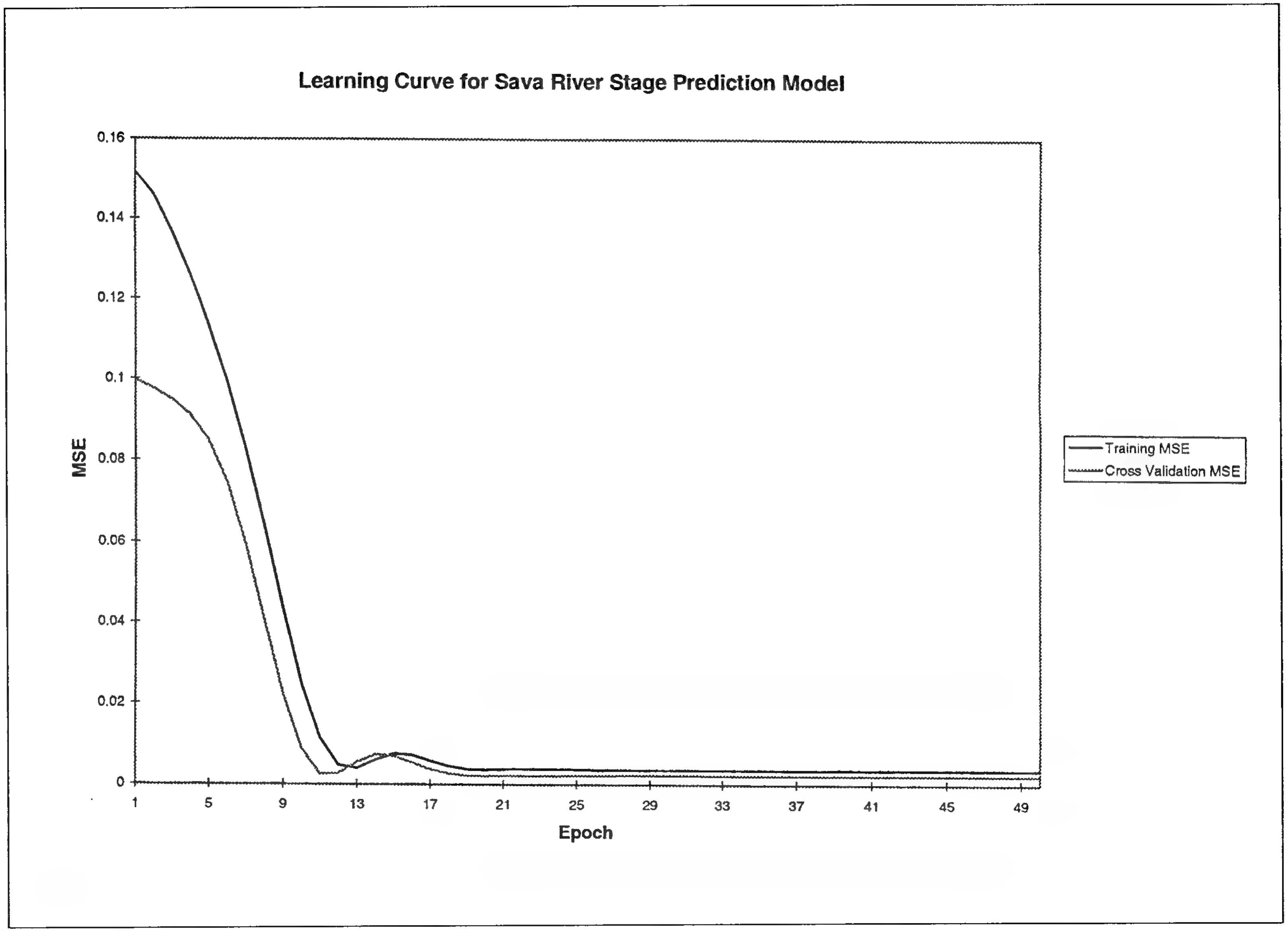


Figure 10. Learning curve for Sava River stage prediction model

Table 1
Performance Analysis of Sava River Stage (m) Prediction Model with MPLs

	Training	Cross-Validation	Testing
MSE	0.0936	0.0601	0.2069
NMSE	0.0183	0.0166	0.0592
MAE	0.2385	0.1894	0.2661
Min Abs Error	0.0005	0.0009	0.0018
Max Abs Error	1.2870	0.8747	2.6596
R	0.9908	0.9931	0.9725

Sava River Daily Flow Prediction Model

As described above, a data file for river flow exists only for eight stations along the main portion of the Sava River from upstream to downstream. Station 8 (Slavonski Brod) is the most downstream gage (no flow gauge at Zupanja) and one of two military bridging sites. From the preliminary analysis for river flow distribution, the first four stations show very similar flow patterns. The flow pattern starts to change at station 5 (Ornac) due to merging tributary flow into the Sava River. The flow pattern change even more rapidly from station 5 to station 7 (Davor) due to the more complicated hydrographic conditions and geomorphology.

Only one-year of flow data was available. Therefore, the length of the training set becomes shorter. Thus, it was decided to use two locations as the inputs: one model using station 1 and station 5 with time lags, trying to predict the flow at station 8. Since there is obvious time lag between station 1 and station 5 and station 8, the model input's preparation gets more complicated if the Multilayer Perceptron Algorithm has to be used. This type of algorithm is called a static classifier. It means that the input-output map depends only on the present input. If it needs to process temporal data, then each time the sample has to be fed to a different input. The result is a very large network. The time-lagged recurrent network (TLRN) model can overcome this disadvantage. TLRNs are MLPs extended with short-term memory structures that have local recurrent connections. The TLRN is a very appropriate model for processing temporal information. The operation procedure for TLRN will not discussed here.

The computational results using TLRN for two-inputs/one output river flow system are presented in Figures 14 through 16. While the training (6 month's data) shows a fair agreement, the cross-validation data (1 month) and testing data (5 months) overestimate the results with some degree of deviation.

The explanation for the difference in the results is: the first 6 months' flow patterns are quite different from the last six month patterns and the record for training is not long enough to adjust the difference from station 1 and 5 to station 8. In addition, the period of hydrologic cycle is an annual event. In order for an ANN to learn different patterns, a multiple annual cycle data would be more beneficial. Some improvement was found if the input series also included station 7. Figures 17 through 19 show the revised model results.

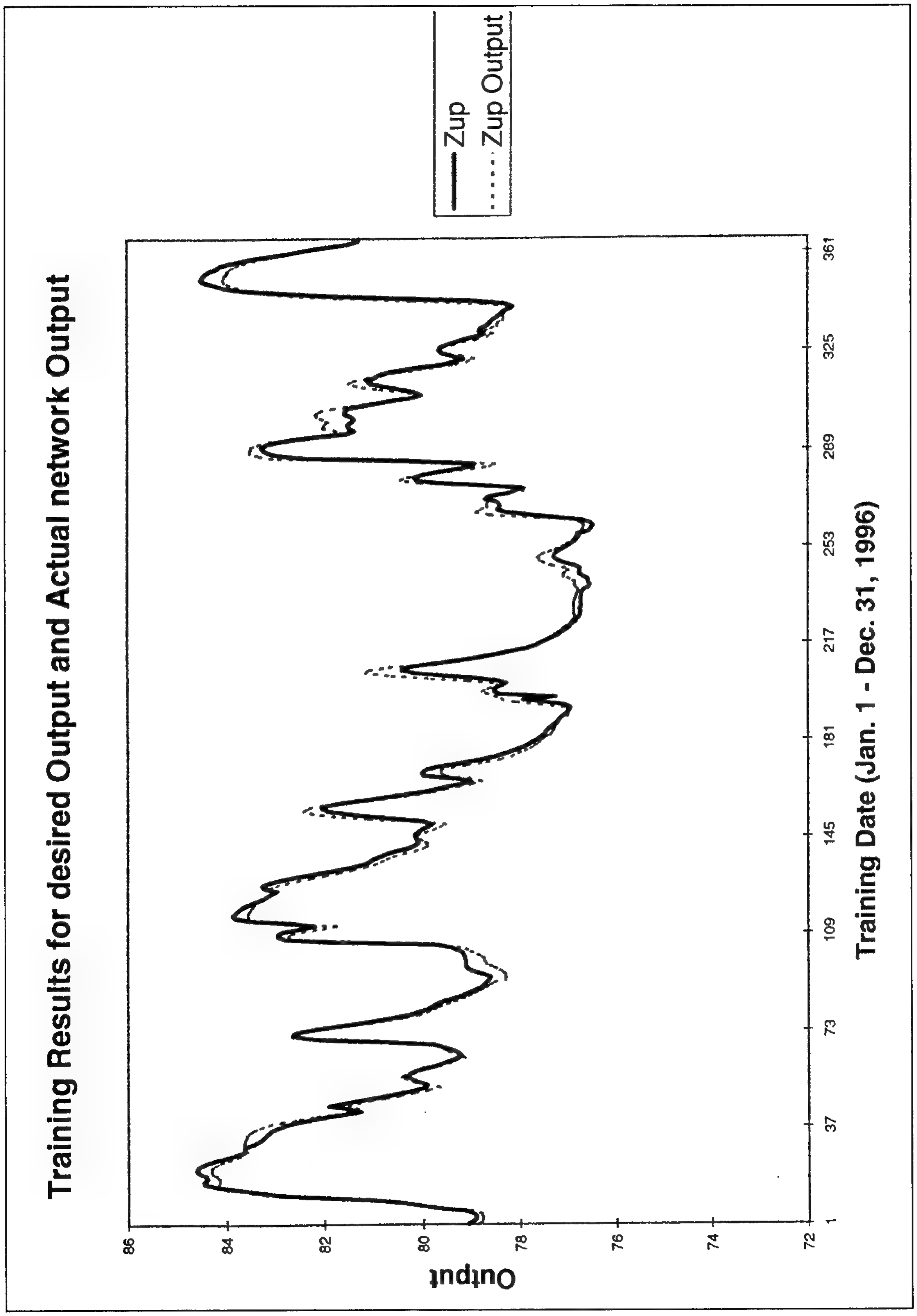


Figure 11. Training results for Sava River stage (m) prediction model (actual (dashed line)) versus model (solid line))

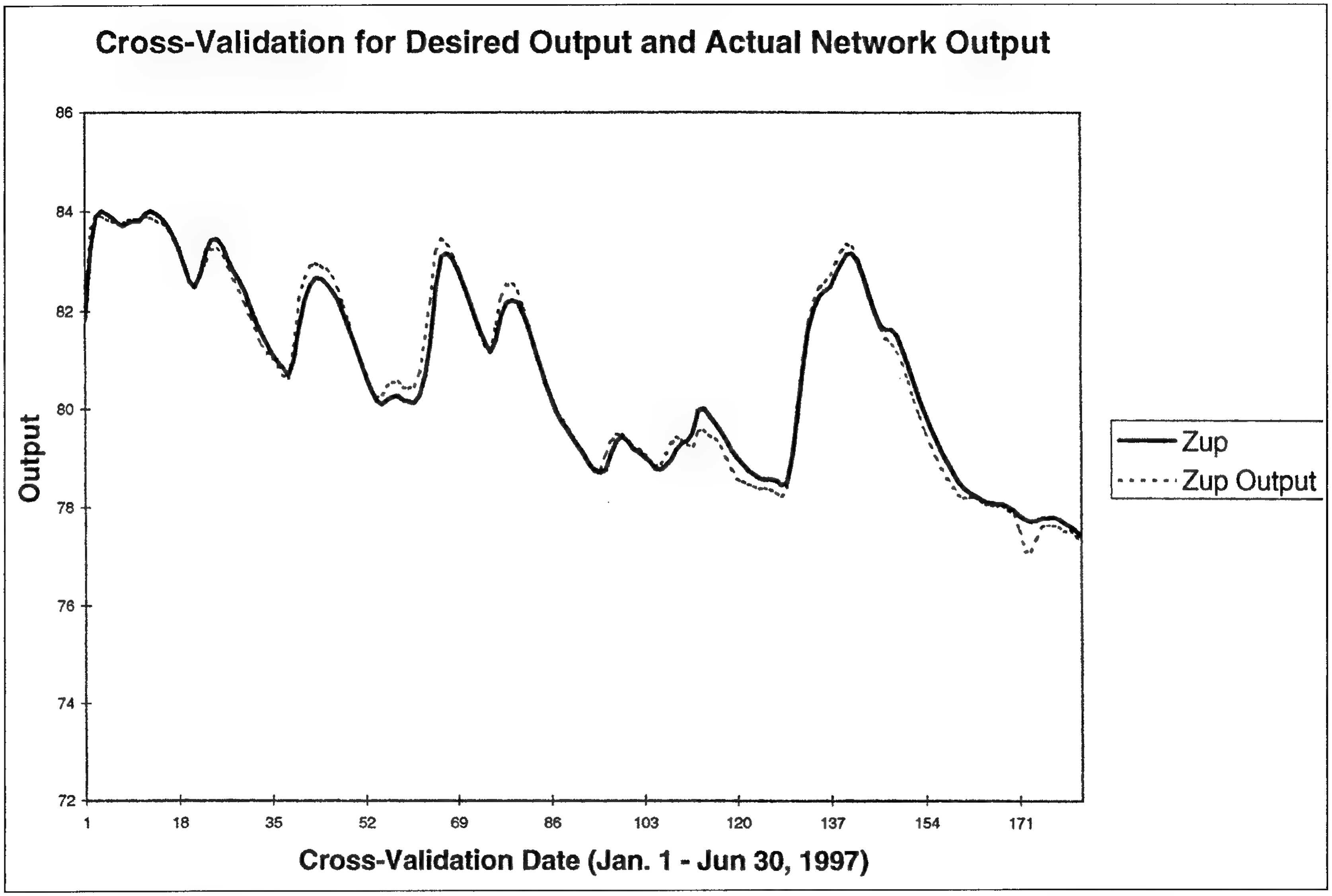


Figure 12. Cross-Validation results for Sava River stage (m) prediction model (actual (dashed line)) versus model (solid line))

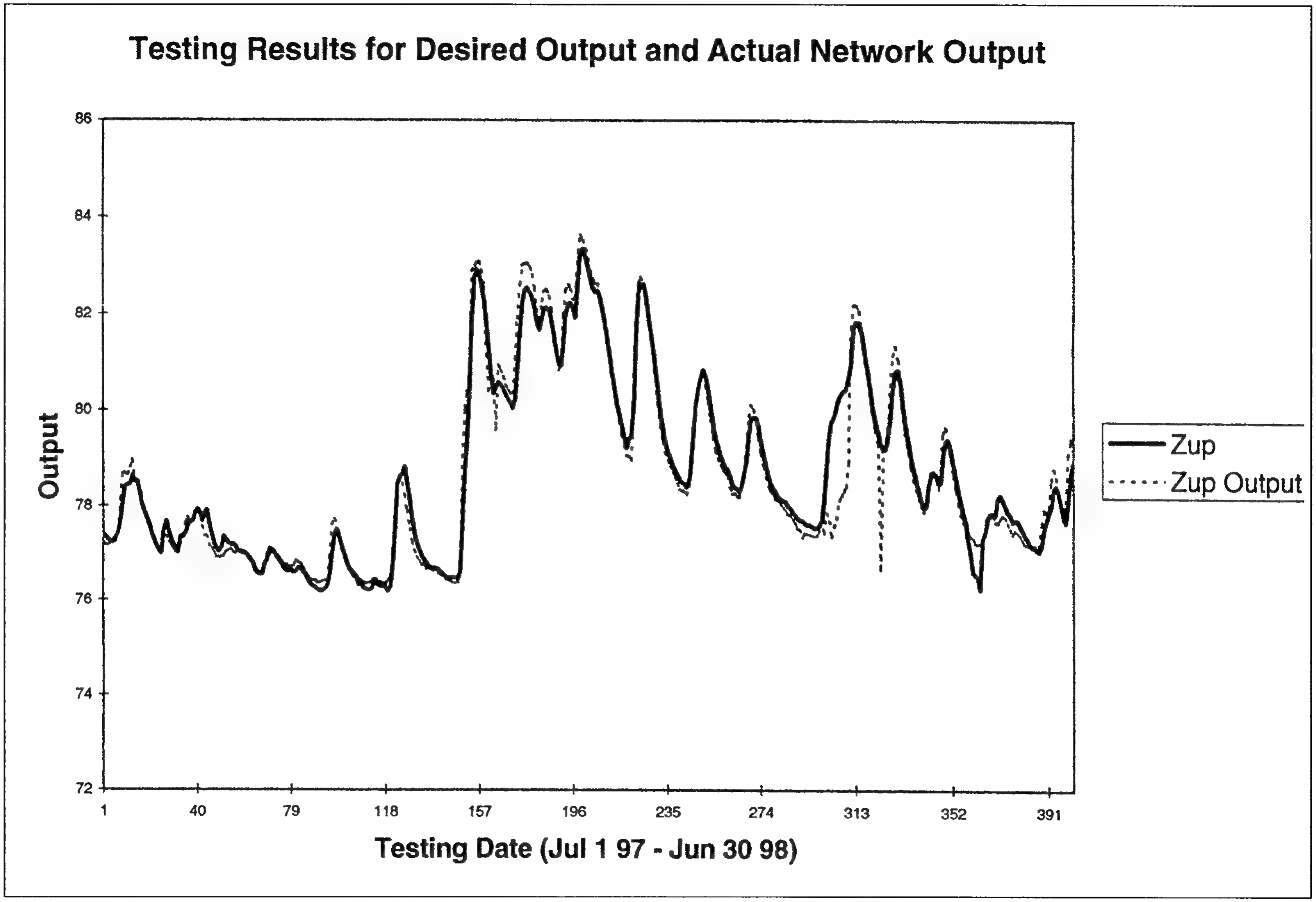


Figure 13. Testing results for Sava River stage (m) prediction model (actual (dashed line)) versus model (solid line))

Table 2
Performance Analysis of Sava River Stage Prediction

	Training	Cross-Validation	Testing
MSE	0.2400	0.4362	0.6714
NMSE	0.0682	0.1241	0.1913
MAE	0.2801	0.3922	0.5236
Min Abs Error	0.0019	0.0005	0.0005
Max Abs Error	3.2841	4.6113	5.1429
R	0.9708	0.9449	0.9115

Sava River Monthly Flow Prediction Model

With the modeling experience regarding the flow prediction described above, it is interesting to investigate the ANN performance with different time scales. Presently, there exists forty years of historical (1926-1965) monthly mean flow data for the Sava River. The TLRN was used to establish an ANN model segmenting the forty years of data as follows: First fifteen years data (1926-1940) as the training set, the next ten years data (1941-1950) as the cross-validation set, and the last fifteen years data (1951-1965) as the testing set. This model used gage stations' 1 (Radece), 5 (Jasenov) and 7 (Slavonski Brod) as inputs and station 8 (Zupanja) as the output. Results: a good agreement (0.0208 meter NMSE and 0.9898 correlation coefficient) was found in the testing performance.

A second scenario was investigated. Here, station 7 was disregarded as one of the inputs and the performance was investigated. The results were surprisingly satisfactory. The correlation coefficient did slightly drop to 0.9778. Figures 20 through 22 show the performance of the 3 learning stages.

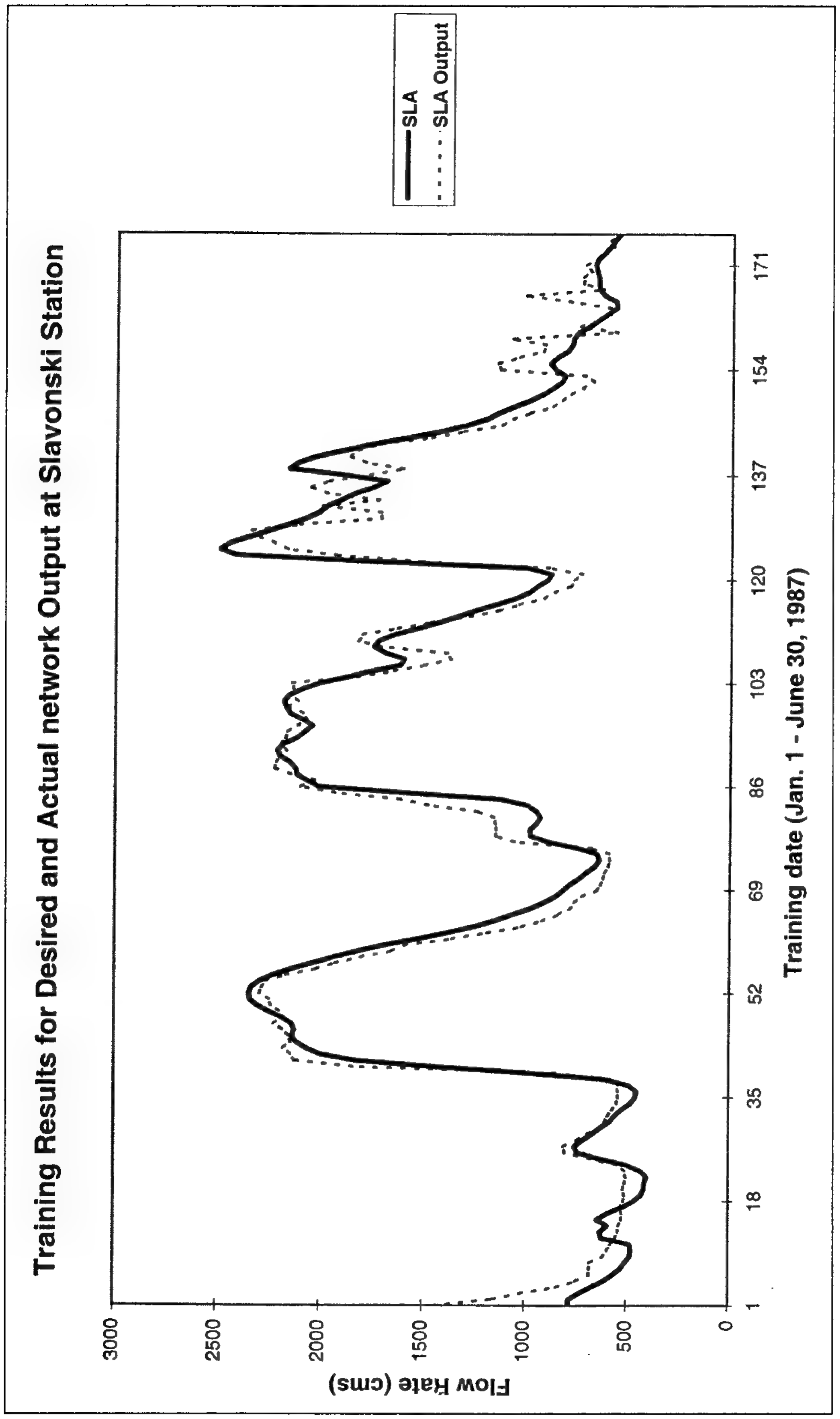


Figure 14. Training results for Sava River daily flow (cms) prediction model with two-inputs/one output system (actual (dashed line)) versus model (solid line))

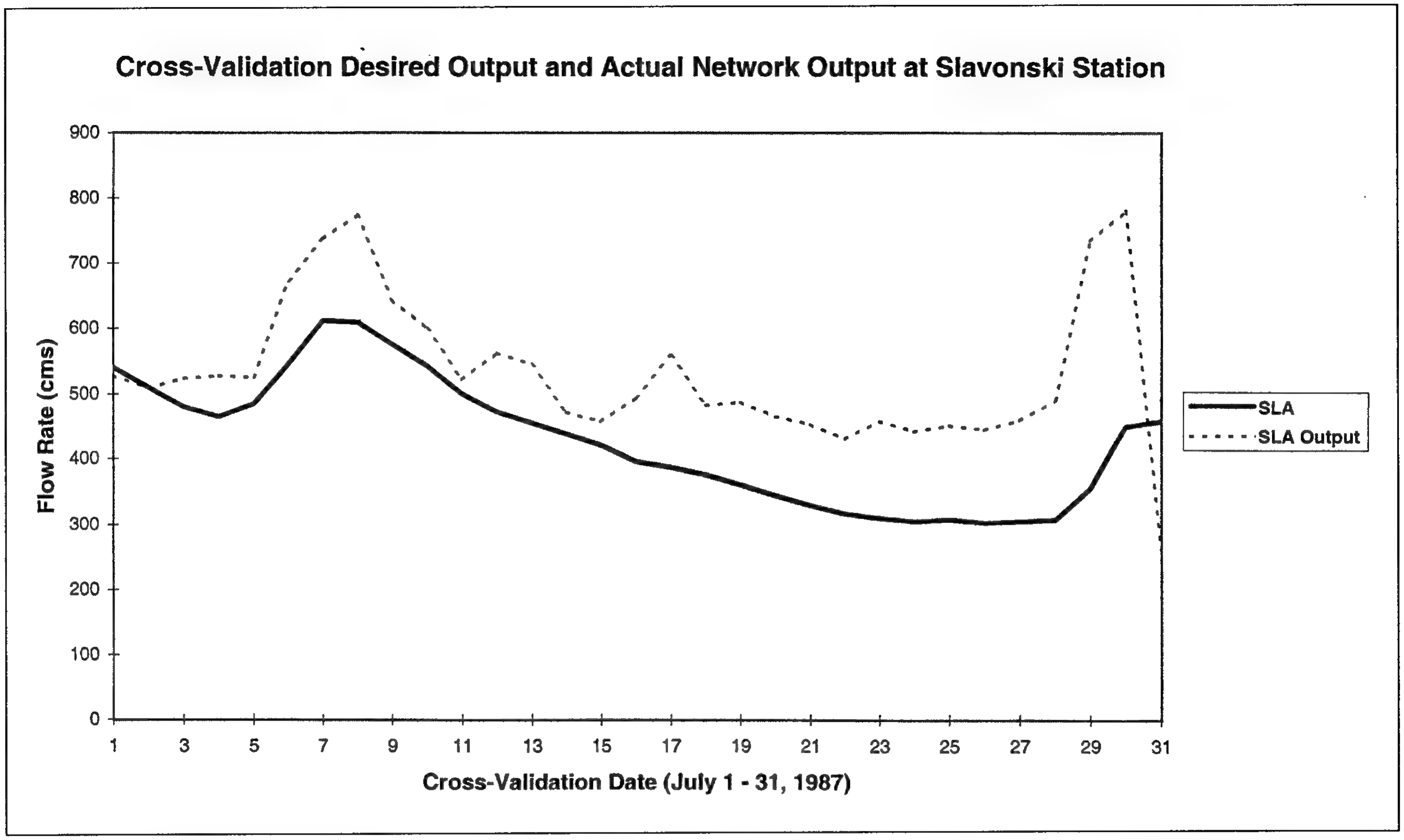


Figure 15. Cross-Validation results for Sava River daily flow (cms) prediction model with two-inputs/one output system (actual (dashed line)) versus model (solid line))

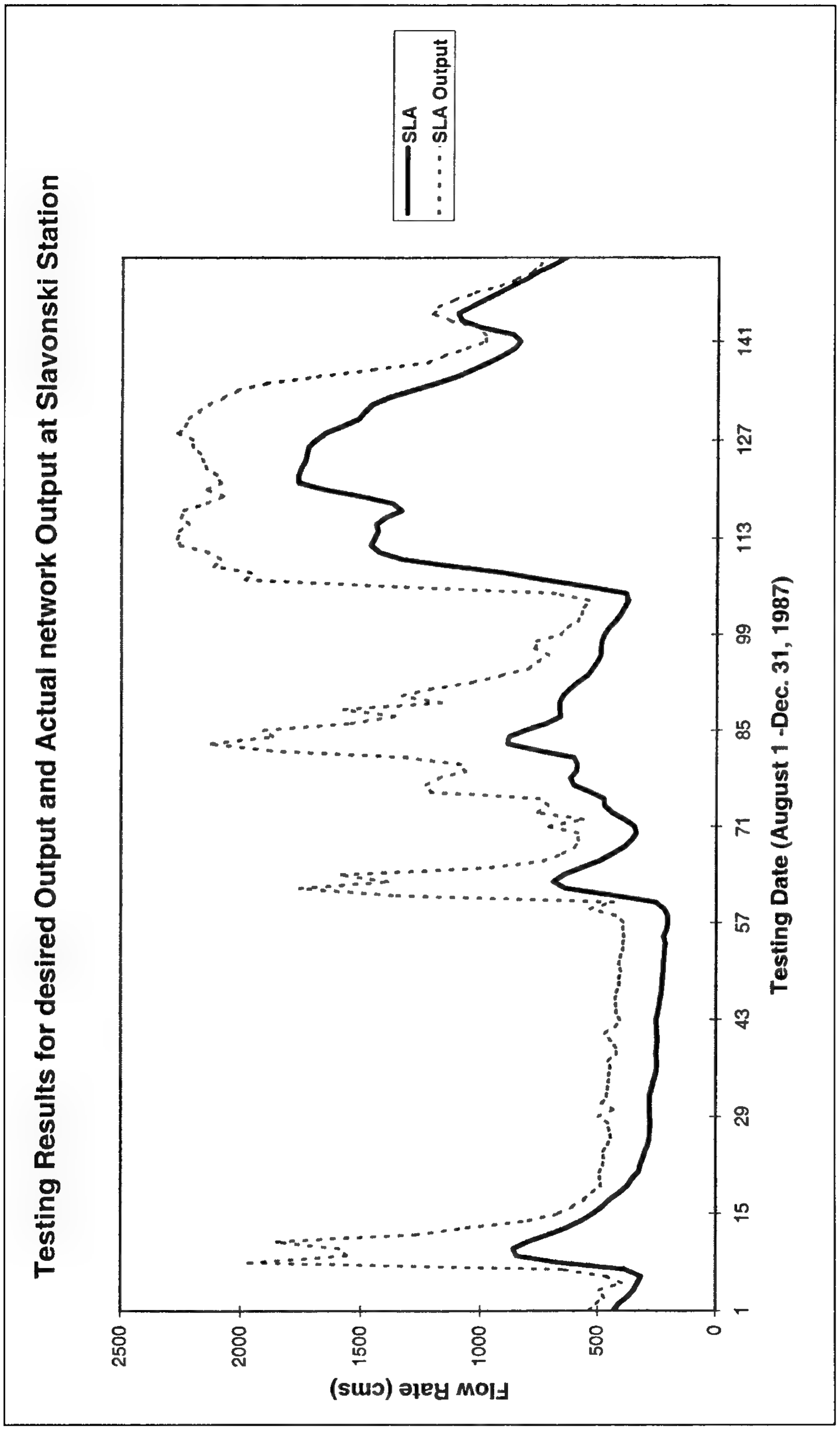


Figure 16. Testing results for Sava River daily flow (cms) prediction model with two-inputs/one output system (actual (dashed line)) versus model (solid line))

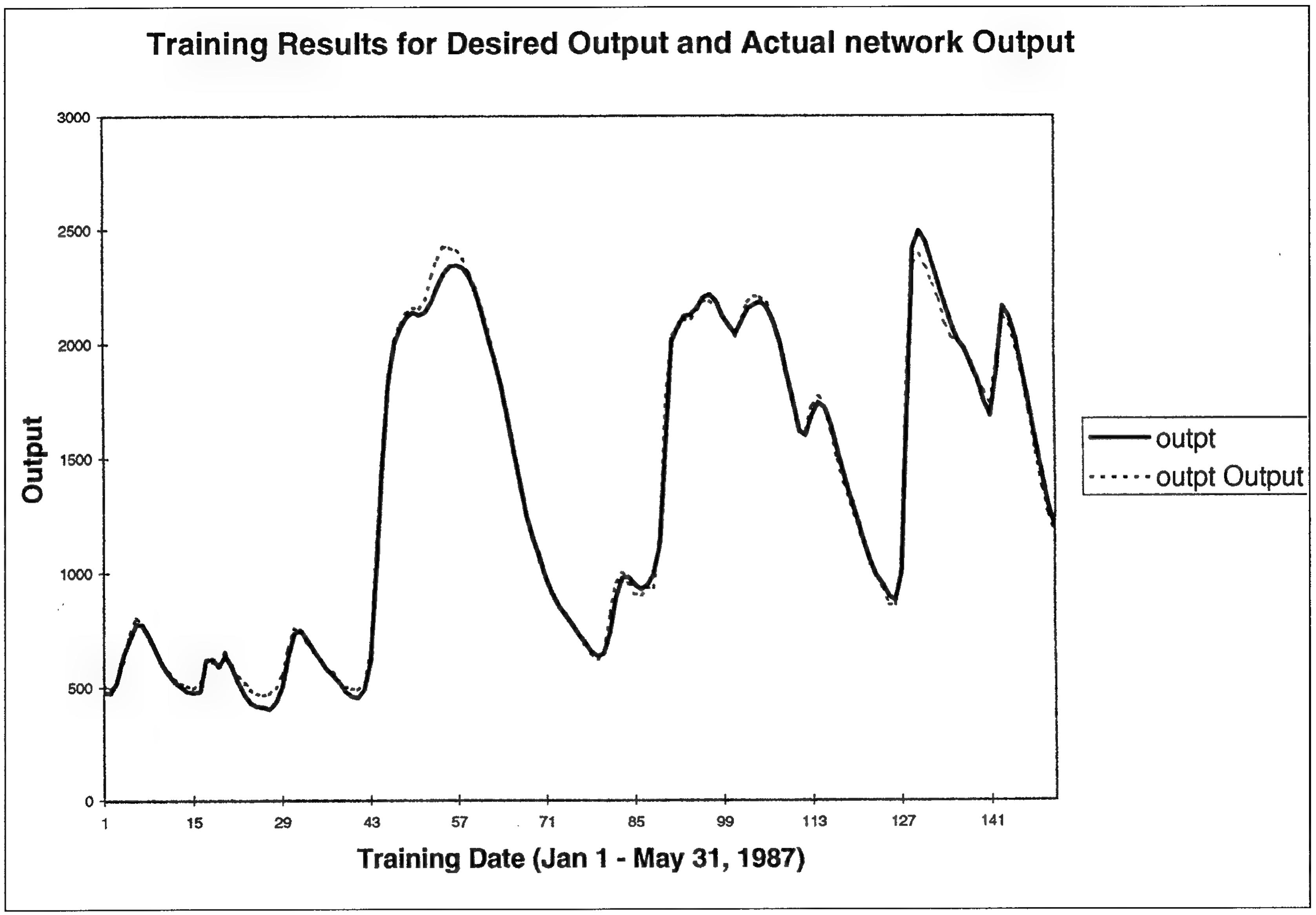


Figure 17. Training results for Sava River daily flow (cms) prediction model with three-inputs/one output system (actual (dashed line)) versus model (solid line))

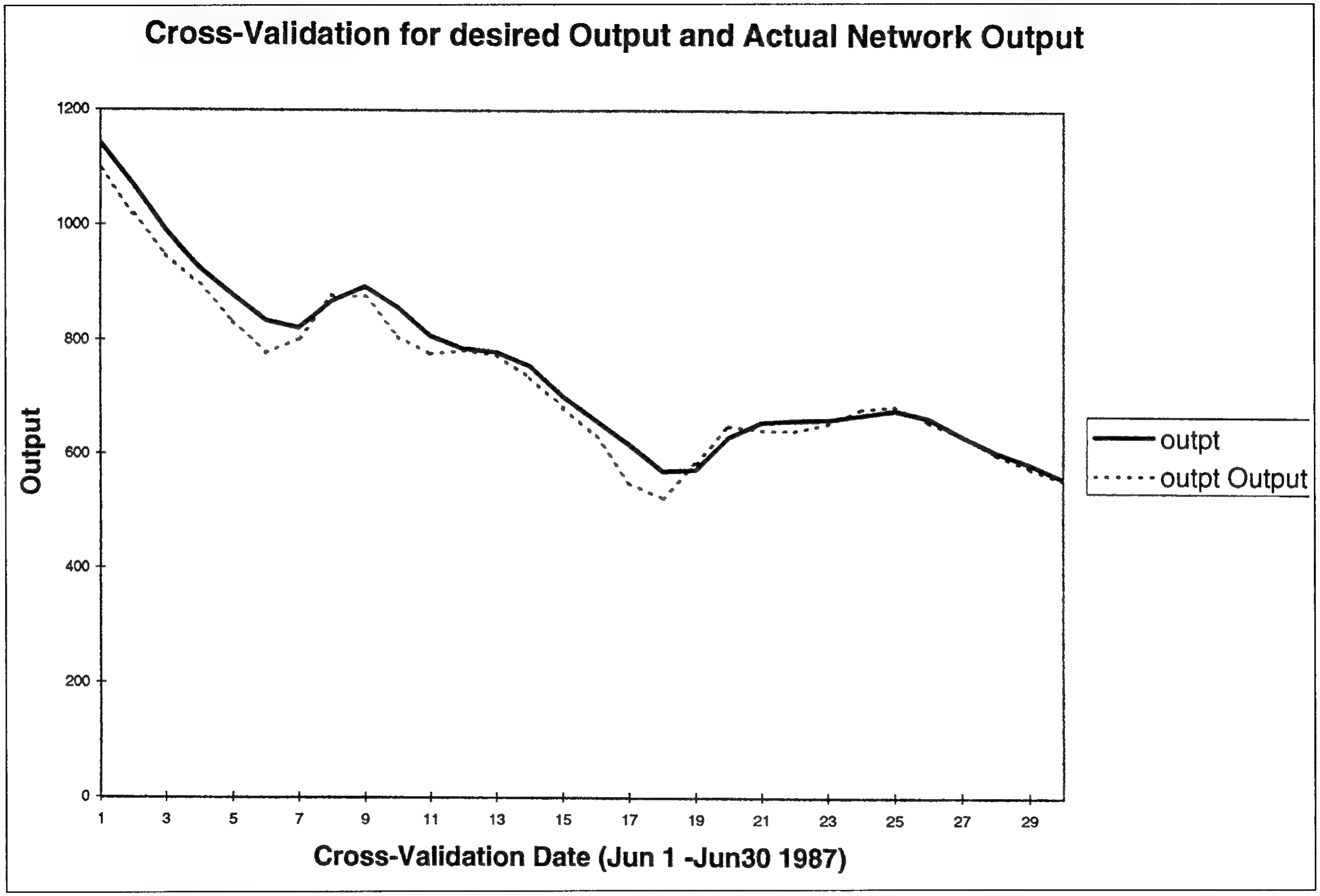


Figure 18. Cross-Validation results for Sava River daily flow (cms) prediction model with three-inputs/one output system (actual (dashed line)) versus model (solid line))

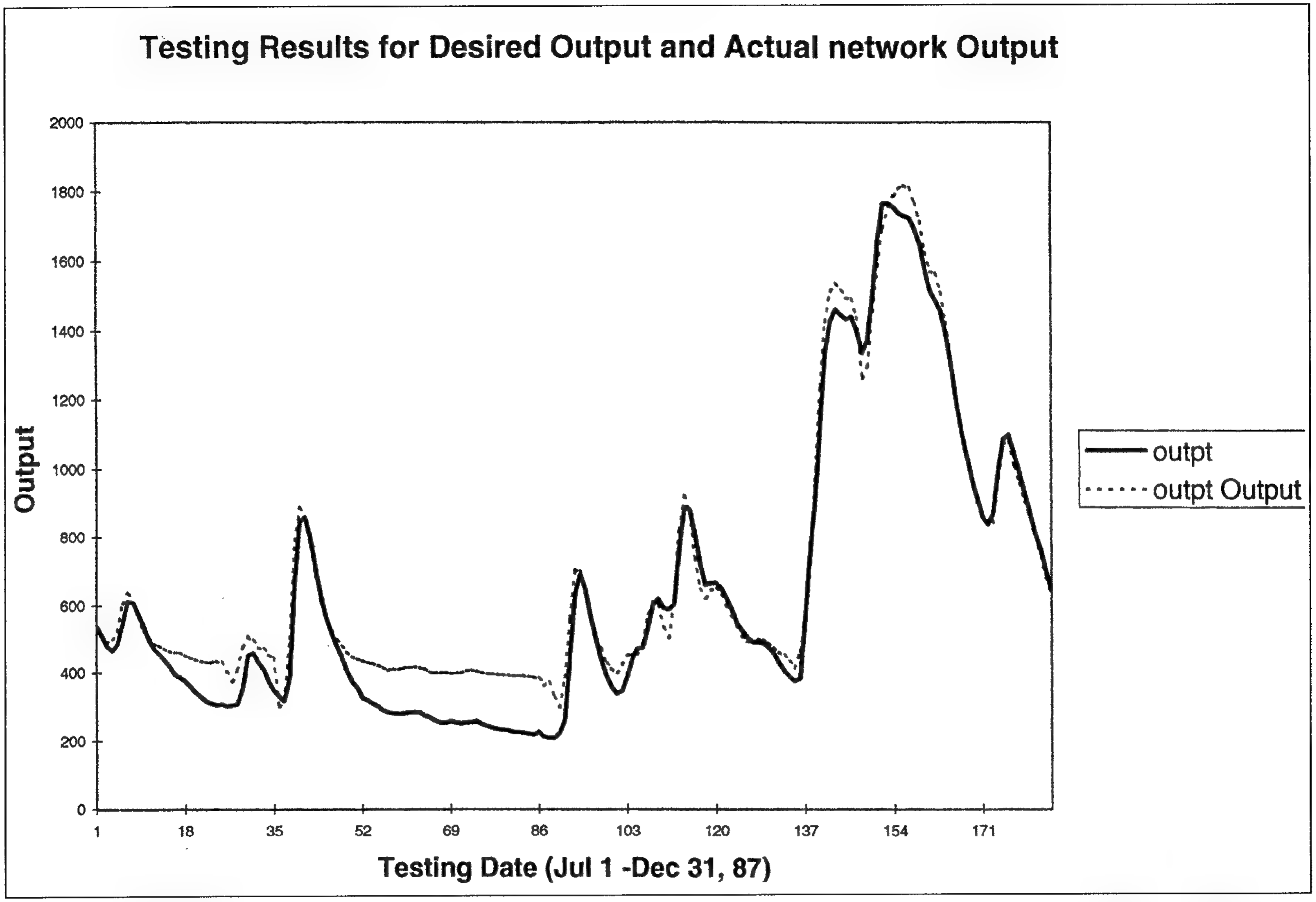


Figure 19. Testing results for Sava River daily flow (cms) prediction model with three -inputs/one output system (actual (dashed line)) versus model (solid line))

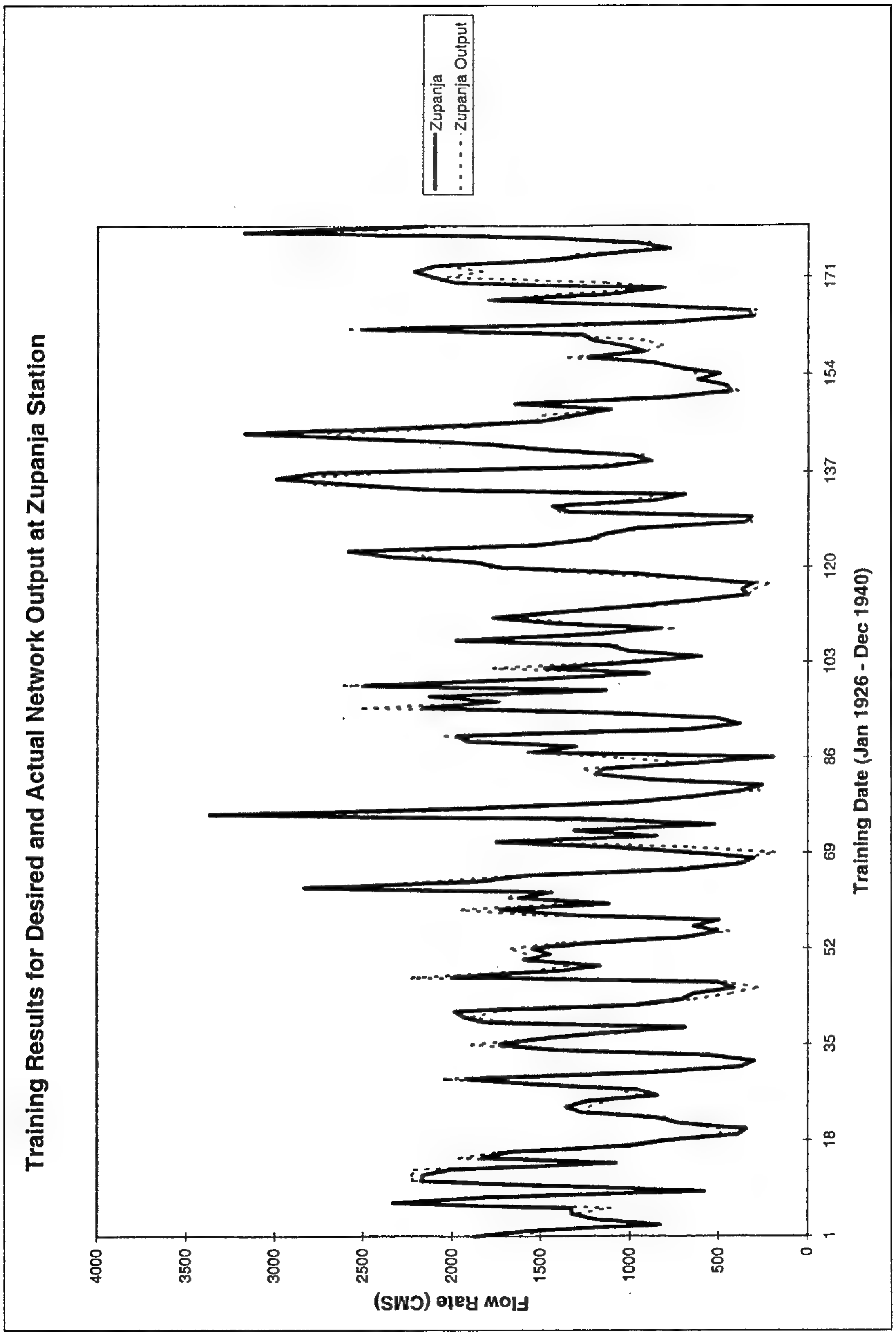


Figure 20. Training results for Sava River monthly flow (cms) prediction model with two-inputs/one output system (actual (dashed line)) versus model (solid line))

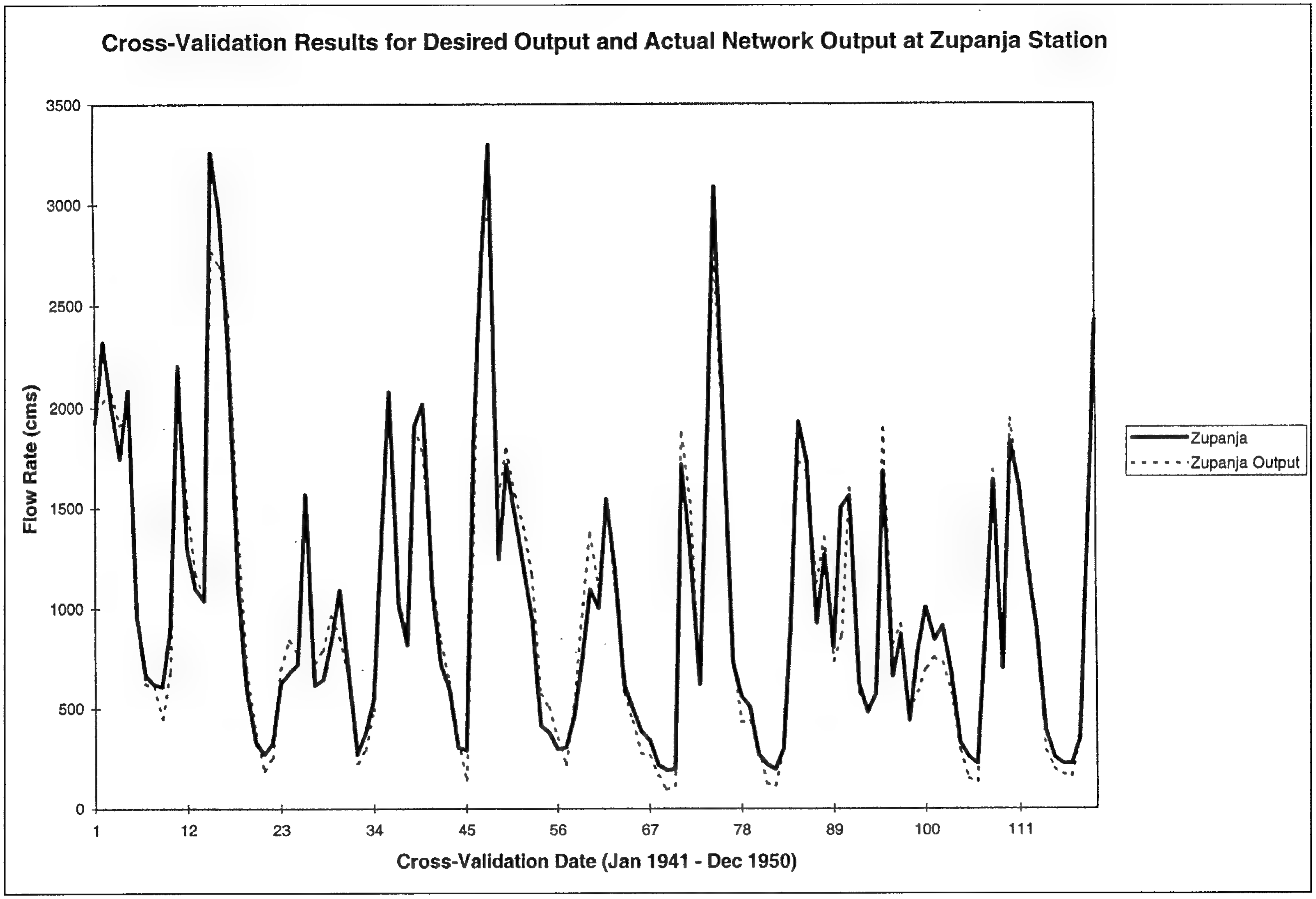


Figure 21. Cross-Validation results for Sava River monthly flow (cms) prediction model with two-inputs/one output system (actual (dashed line)) versus model (solid line))

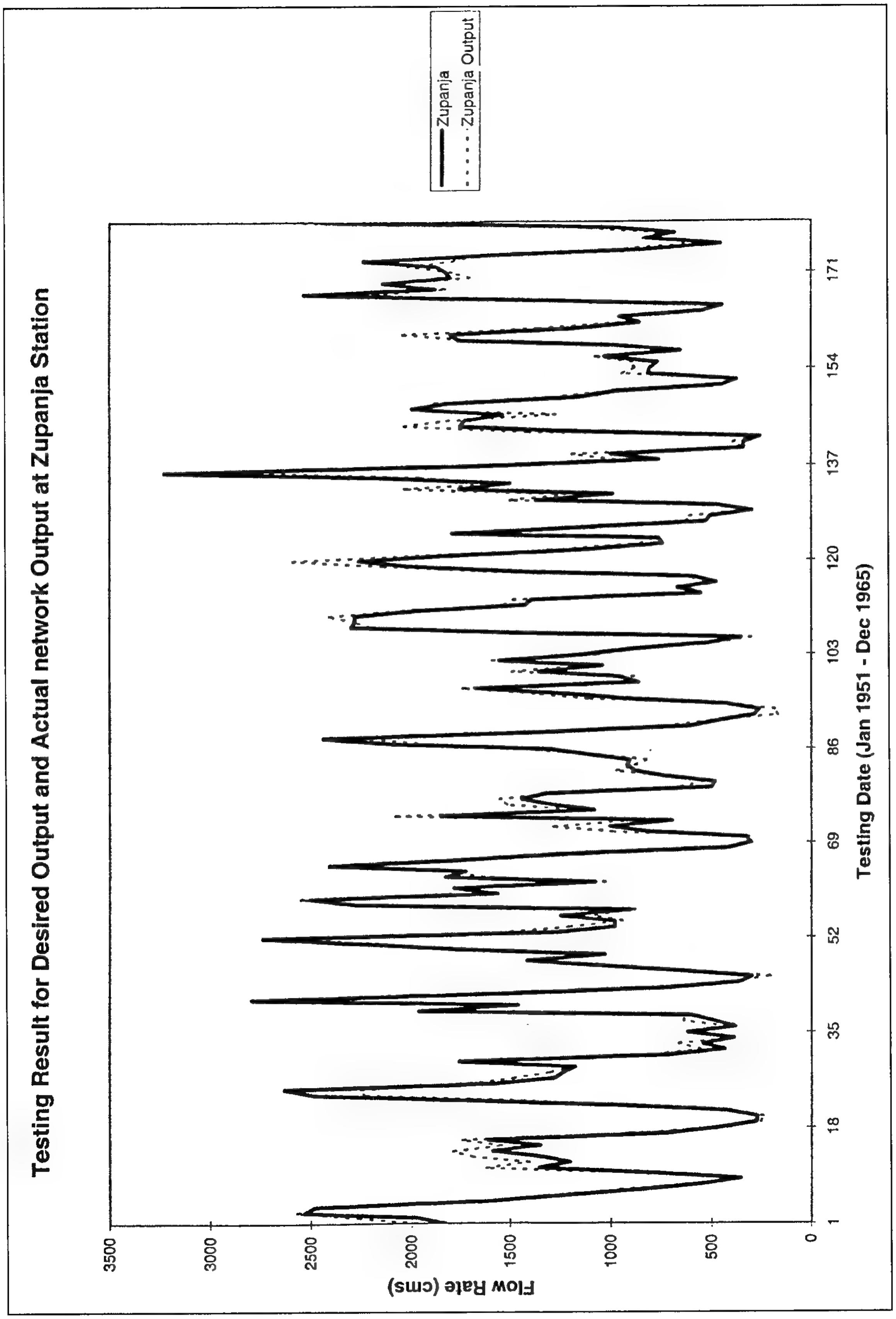


Figure 22. Testing results for Sava River monthly flow (cms) prediction model with two-inputs/one output system (actual (dashed line)) versus model (solid line))

4 A Mississippi River Segment Application

The Mississippi River Basin (Figure 23) is the largest watershed in the United States. Understanding the dynamic changes of this river flow system is a subject of nationwide concern. In this chapter, a rainfall-runoff model using ANNs was incorporated in a segment of the lower portion of the Mississippi River. The purpose was to compare the diverse flexibility of ANNs in its ability to test, analyze and predict hydrological characteristics of a larger, more data rich, watershed compared to our earlier analysis with the Sava River Basin. Figure 24, below, shows the segment of the lower Mississippi River that was used in this study.

Watershed Description

The Lower Mississippi River is considered to begin at Cairo, IL at the confluence of the Ohio and Middle Mississippi Rivers. It travels southward a distance of approximately 954 miles emptying in the Gulf of Mexico at Head of Passes, LA. In 1973, a serious flood occurred in the Lower Mississippi River. The peak flows for the crest stages were over 1.5 million cfs. Major flooding that occurred during that time showed the need for a system that better forecasts river stage/ river flow characteristics. This could prove vital in the future to serve as an essential tool in reducing flood damage through better forecasting systems if these conditions would happen again.

In this study, ANN was used to predict the river flow at Memphis, TN, from the upstream gage at Thebes, IL, near the confluence with the Ohio River. The gage station at Metropolis, Illinois provided characteristic flow of the Ohio River. Further downstream, we considered the lateral contributions of the tributaries: Obion, Hatchie, Loosahatchie, and Wolf Rivers in West Tennessee, as well as precipitation influence throughout this river basin. The combined flow in the Mississippi River below the confluence point is approximately 55 percent out of the Ohio River and 45 percent out of the Mississippi River main stem. Therefore, the purpose of this study was to identify the prediction capability with minimum hydrologic information using ANNs to determine the contribution of the Ohio River to flooding portions of the lower Mississippi River Basin. Under this consideration, two upstream river flow gages were used as the model input instead of using one combined river flow that represented both input values from

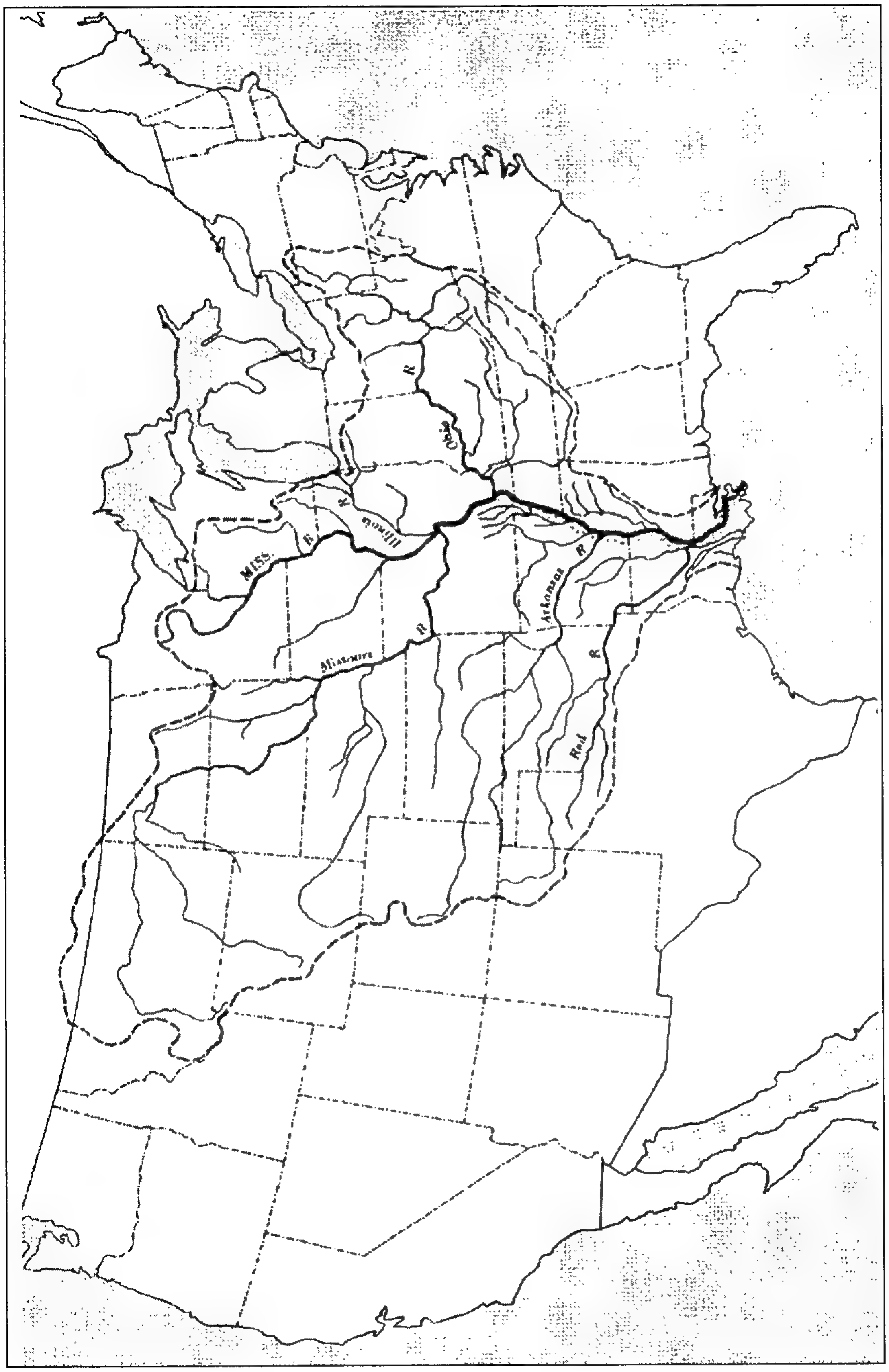


Figure 23. Mississippi River Basin

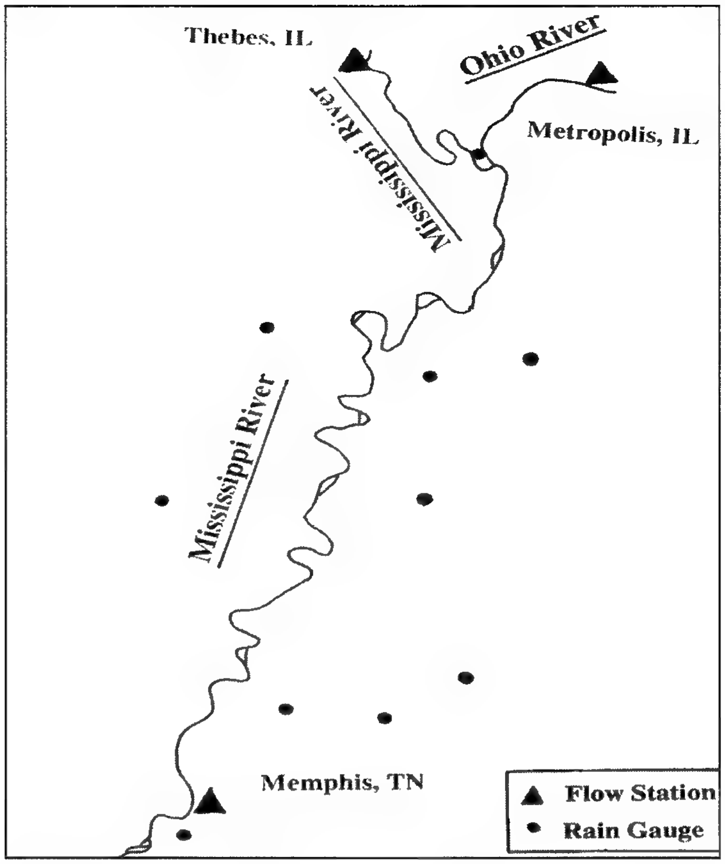


Figure 24. A Mississippi River segment watershed

the Middle Mississippi and Ohio Rivers. This was essential to further examine attributions in our ANN modeling by both river systems prior to the flow confluence.

Database Development and Preliminary Data Analysis

A database was developed using 16 years (from 1975 to 1990) of daily river flow data from three major stations Thebes, Illinois (Upper portion of Mississippi River), Metropolis, Illinois (Ohio River), and Memphis, Tennessee; (Figure 25); four river tributaries in Western Tennessee using the closet gage reading to the Mississippi River on each tributary; (Figure 26). Some of the data had missing values, ranging from several months to a two-year lapse. Reconstruction and estimation of these data sets were established by the regression method, drainage area ratio and transfer function model using ANN. Additionally, we applied daily record from ten precipitation collection stations (Figure 27), which were uniformly (approximately selected) distributed over the study portion of the river basin.

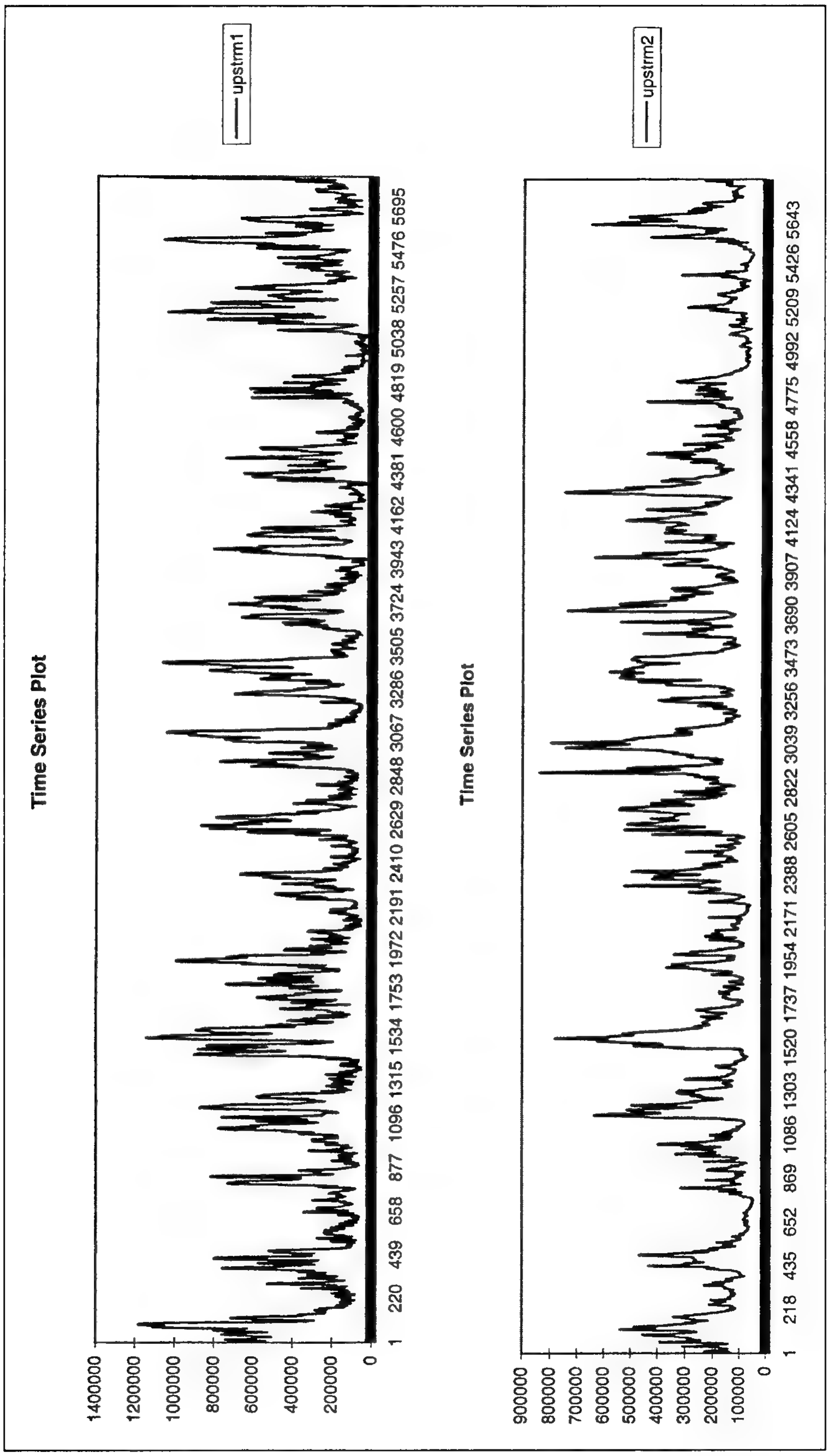


Figure 25a-b. Daily riverflow (cfs) at Thebes and Metropolis stations respectively (1975-90)

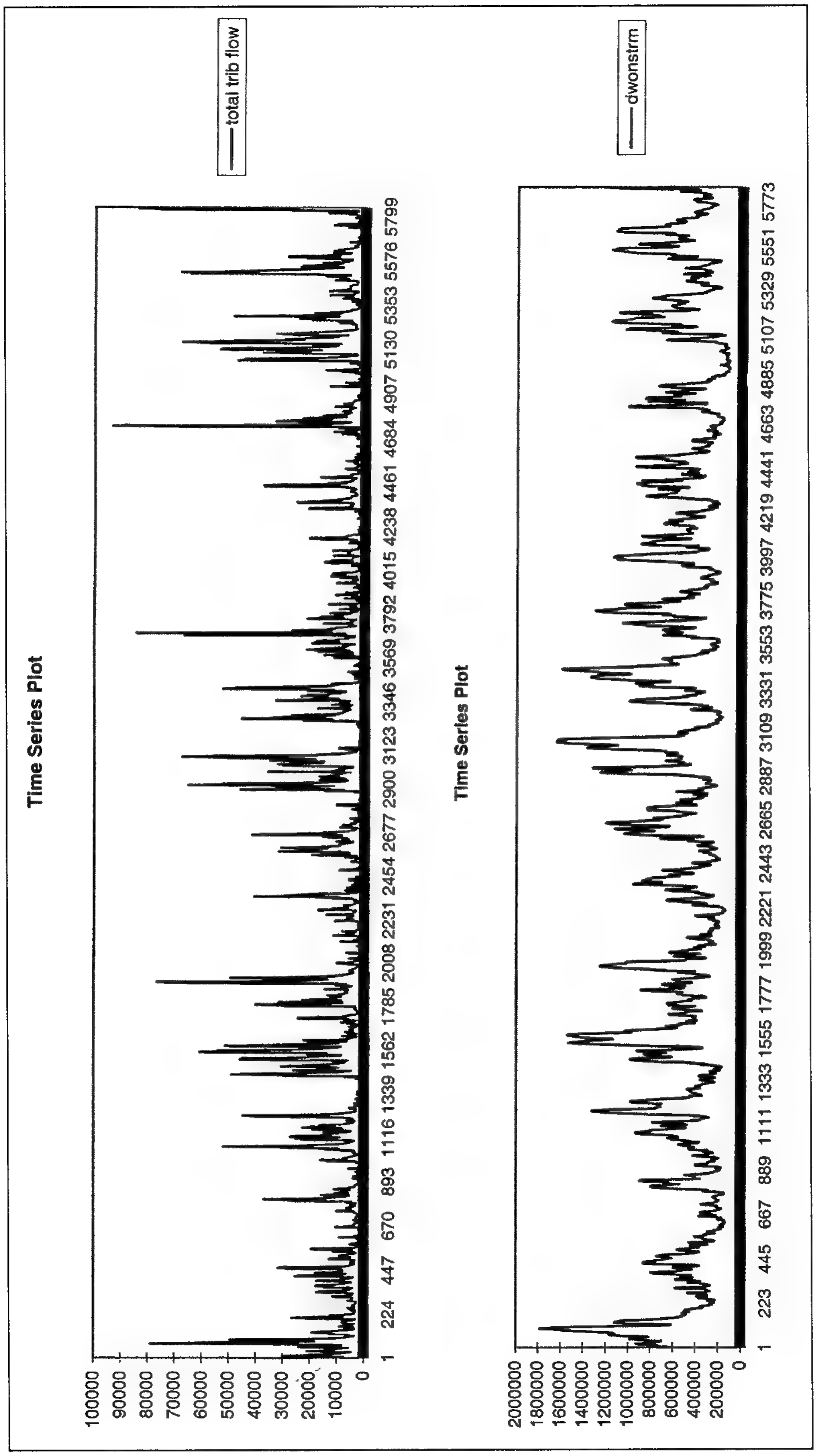


Figure 25c-d. Daily riverflow (cfs) of total tributary and Memphis station respectively (1975-90)

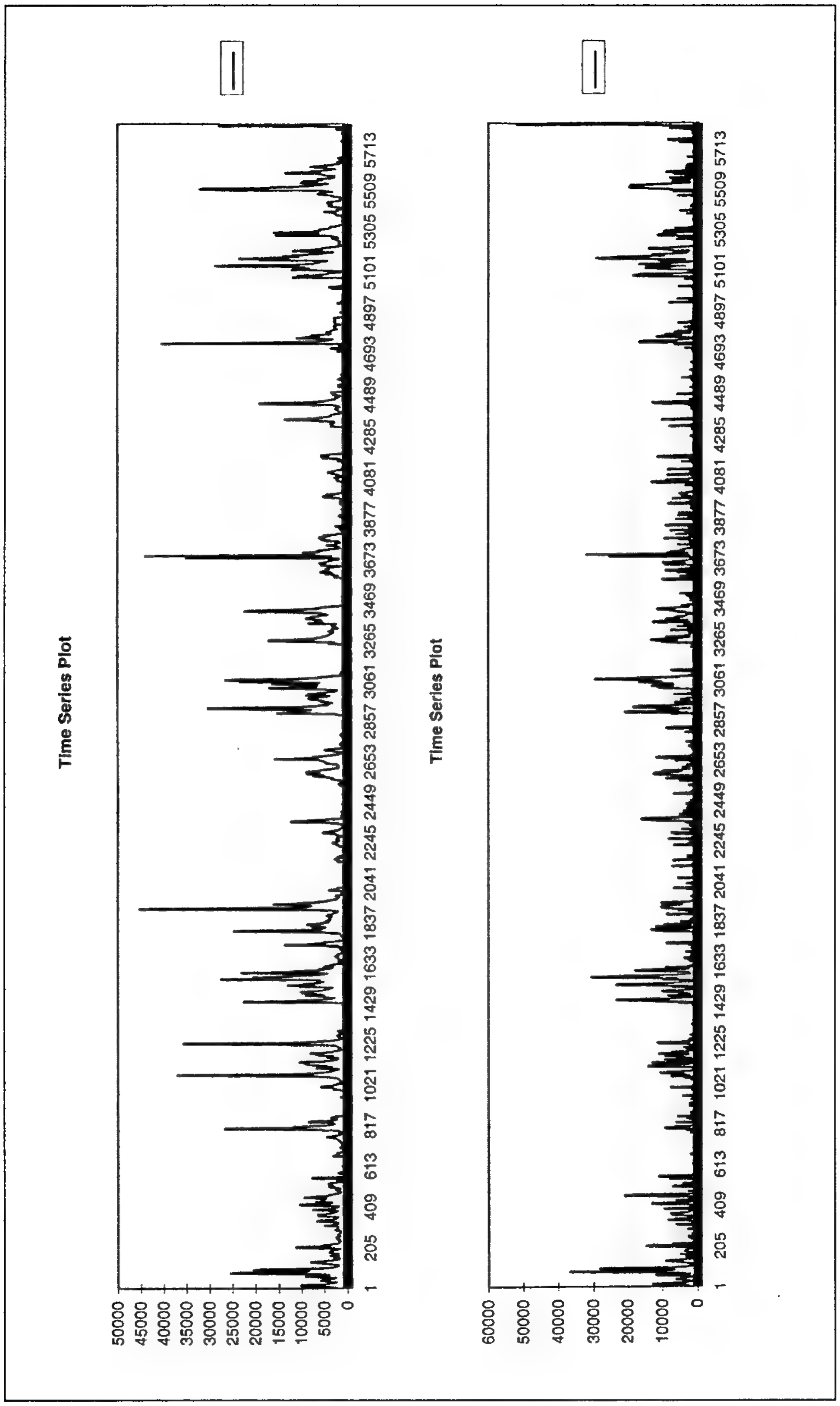


Figure 26a-b. Daily riverflow (cfs) at Wolf and Looschachie rivers respectively (1975-90)

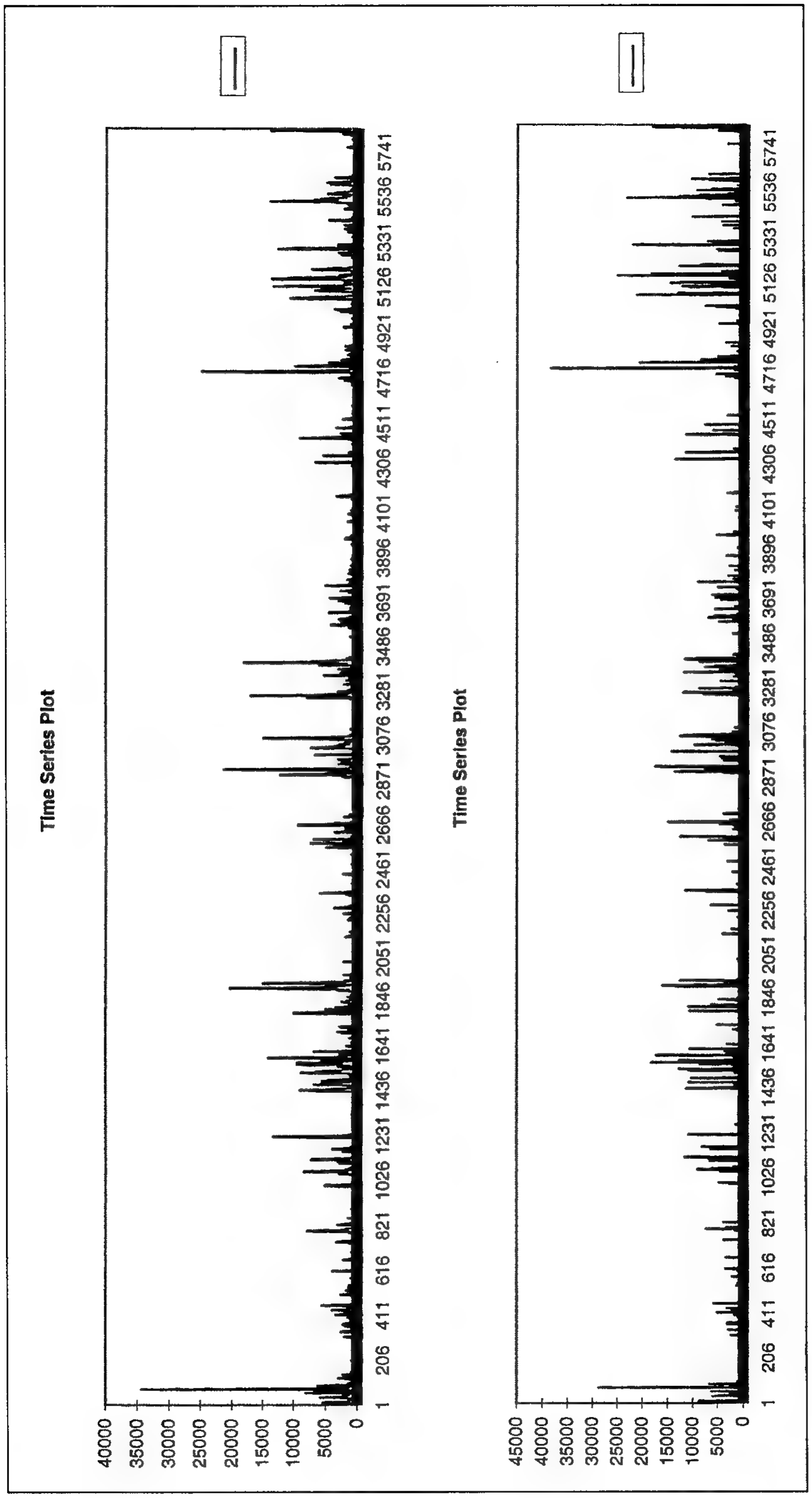


Figure 26c-d. Daily riverflow (cfs) at Hatchie and Obion rivers respectively (1975-90)

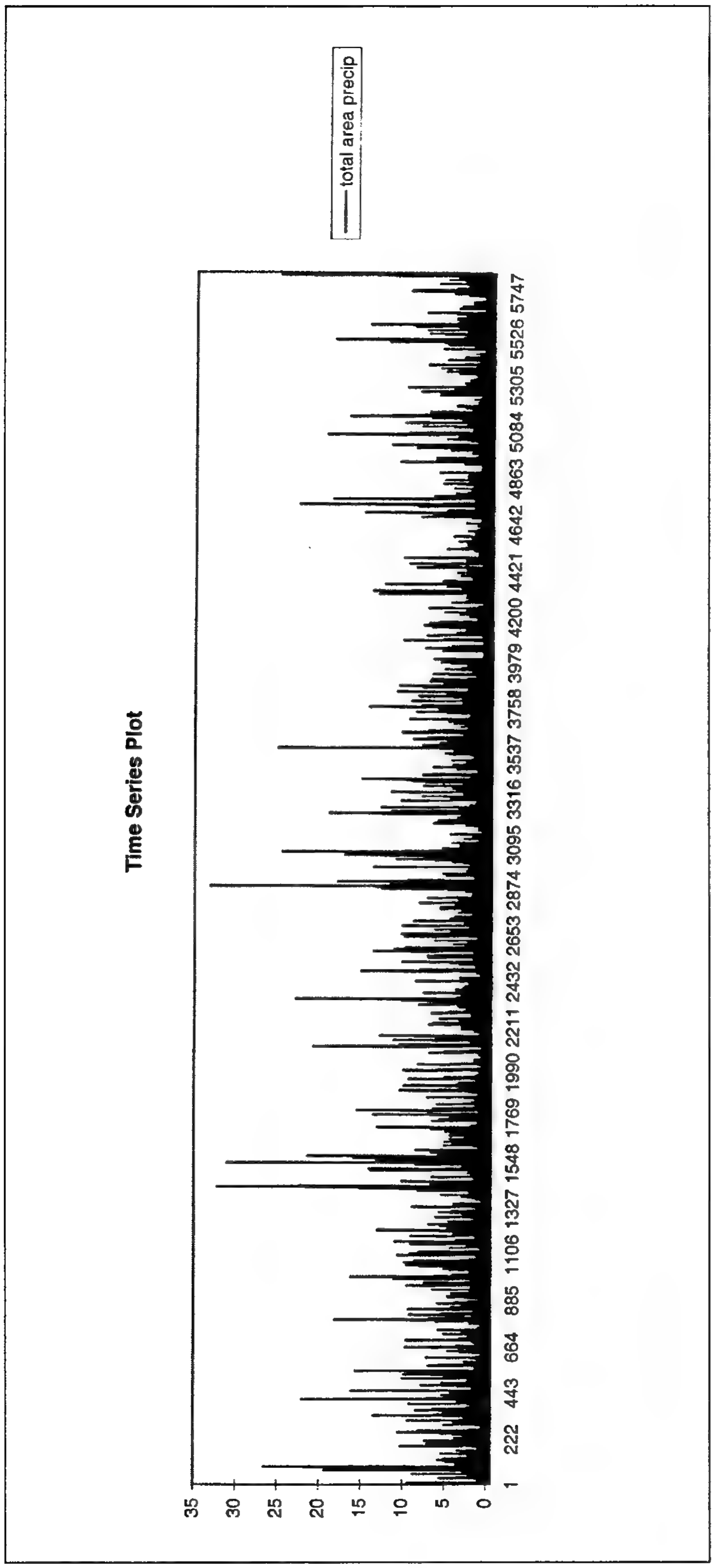


Figure 27. Total daily precipitation (inches) of ten rain-fall gauges in a segment of Lower Mississippi River basin (1975-90)

A preliminary data analysis was conducted by a series of X-Y plots between the downstream flow and possible inputs. Figure 28 examined the relationship between each tributary and the downstream flow patterns. While the Obion River shows some contribution to the flow rate at Memphis, the Loosahatchie, Hatchie and Wolf Rivers are not strongly related it. Since this correlation is highly nonlinear due to the hydrologic processes and time-lagged for signal travel, it was determined to combine all four tributaries as one single lateral contribution. The similar treatment also is applied to the total precipitation. Under this consideration, Figure 29 represents the correlation of four major inputs with the downstream flow as the output. The ranking of data input significance for this study is the contribution of main stem of the Mississippi River at Thebes, Ohio River at Metropolis, total tributary flows, and precipitation values in this region.

ANN Rainfall-Runoff Model

With our study of the Mississippi River, we analyzed different configurations of neural network models, those that varied the input layer structure, to compare numerical results. These results are in the form of statistical parameters. Some of the more common parameters that were analyzed after our input variations were: Correlation Coefficient (CC) and Normalized Mean Squared Error (NMSE). Our approach for this study using ANNs started from the simplest structure (two inputs/one output) to more complicated hydrological systematic model (four input/one output). In other words, a rainfall-runoff model was first constructed using two upstream flows as inputs and the downstream river flow as the output. The first 6 years' of data were used as the training; the next 2 years' data used as the cross-validation set, and the last 8 years used as the testing data. A MLP feed-forward BP architecture design was adopted. Fairly high accuracy, statistically speaking, in comparing the testing sets based on the performance of NMSE and CC was evident. The CC ranges were 0.95, 0.93 and 0.94 for the training, cross-validation and testing sets respectively. However, graphical comparisons show the spikes match very well, but there was a noticeable phase shift that existed between the observed values and simulated outputs. This difference implies the consideration of time lags is required.

The second test of this model was to use the Time Lagged Recurrent Networks (TLRNs). The TLRN approach produced a significant improvement. Excellent agreement for three learning stages is shown in Figures 30, 31 and 32. With an extremely high CC for both training and testing, this indicated that the downstream flow prediction might only require the information in the upper stream gages without the knowledge of the watershed characteristics (e.g. precipitation).

The next scenario of this study was to add the rainfall factor and total tributary flow as the input variable. This gives three variables to the data set input for ANN modeling. While two different types of models consisting of three-input/one output systems (two upstream flow and total tributary; two upstream flow and total rainfall) are easy to converge for the training of a neural network, a model having four-input/one output model has the difficulty for convergence given a LinearAxon output activation function. In spite of changing the activation

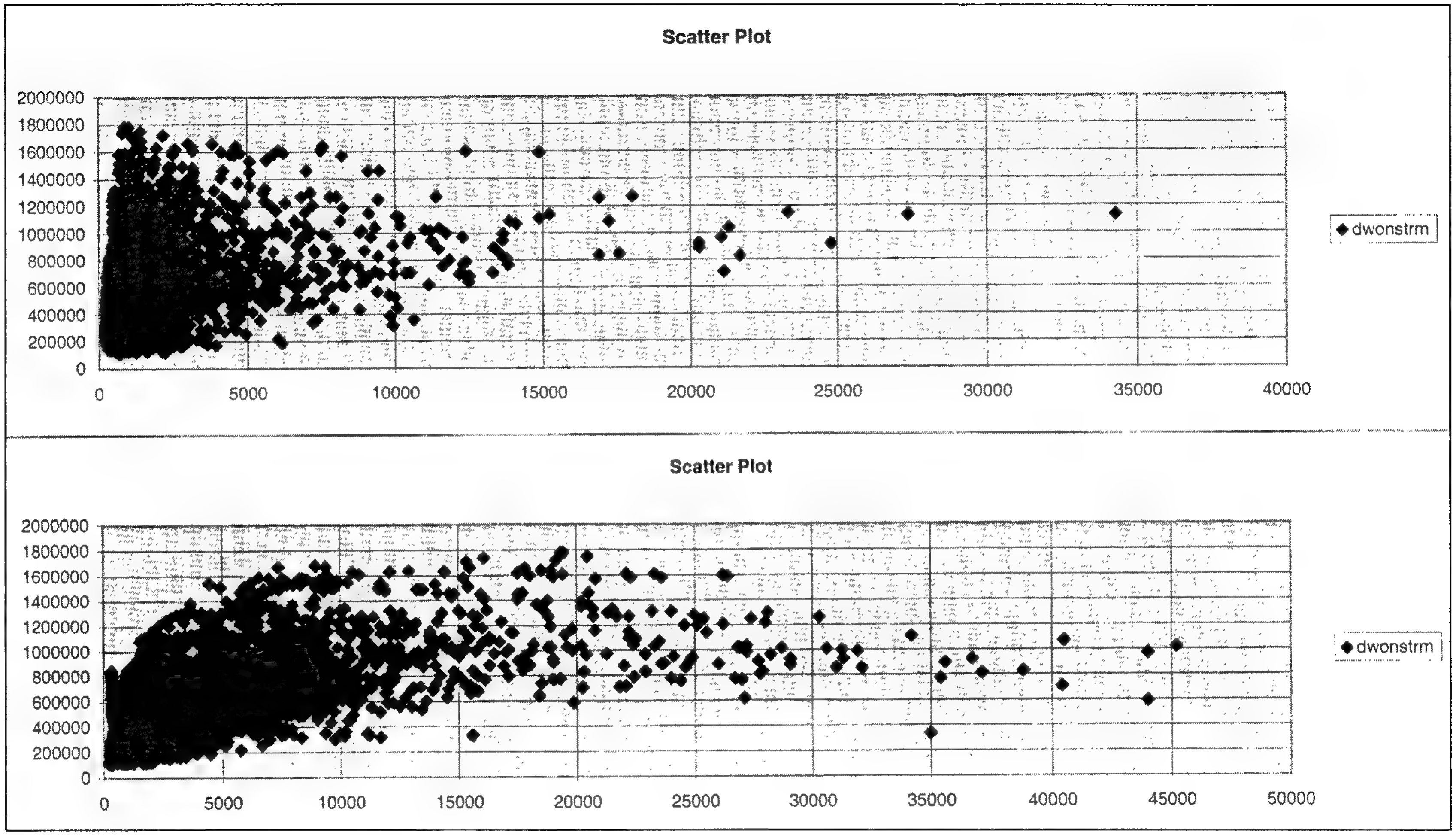


Figure 28a-b. Correlation of riverflow (cfs) between (a) Wolf River and (b) Loosahachie River and Mississippi River (y-axis) at Memphis station respectively

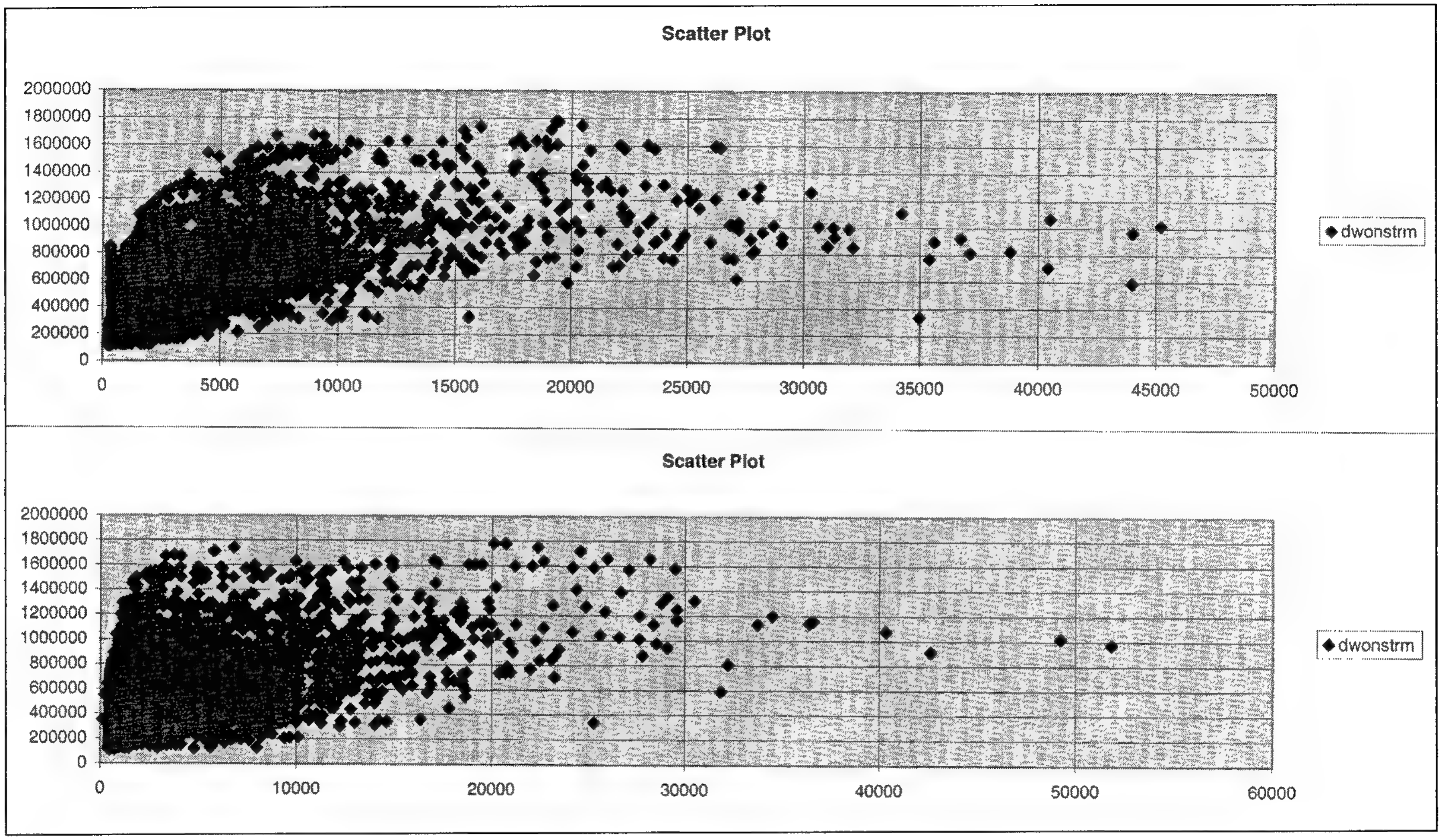


Figure 28c-d. Correlation of riverflow (cfs) between (c) Hatchie River and (d) Obion River and Mississippi River (y-axis) at Memphis station respectively

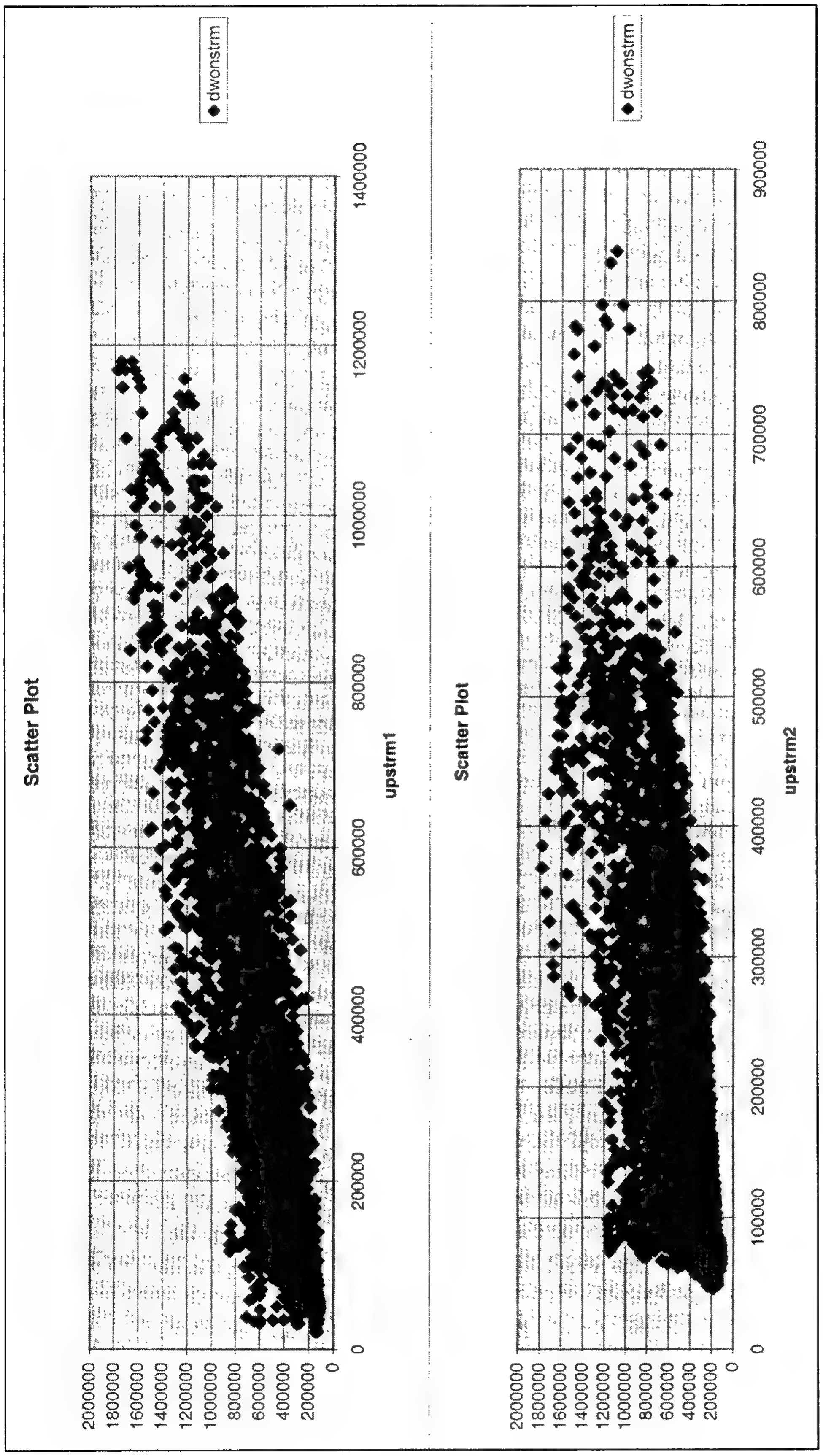


Figure 29a-b. Correlation of riverflow (cfs) between (a) Thebes and (b) Metropolis and Memphis (y-axis) stations respectively

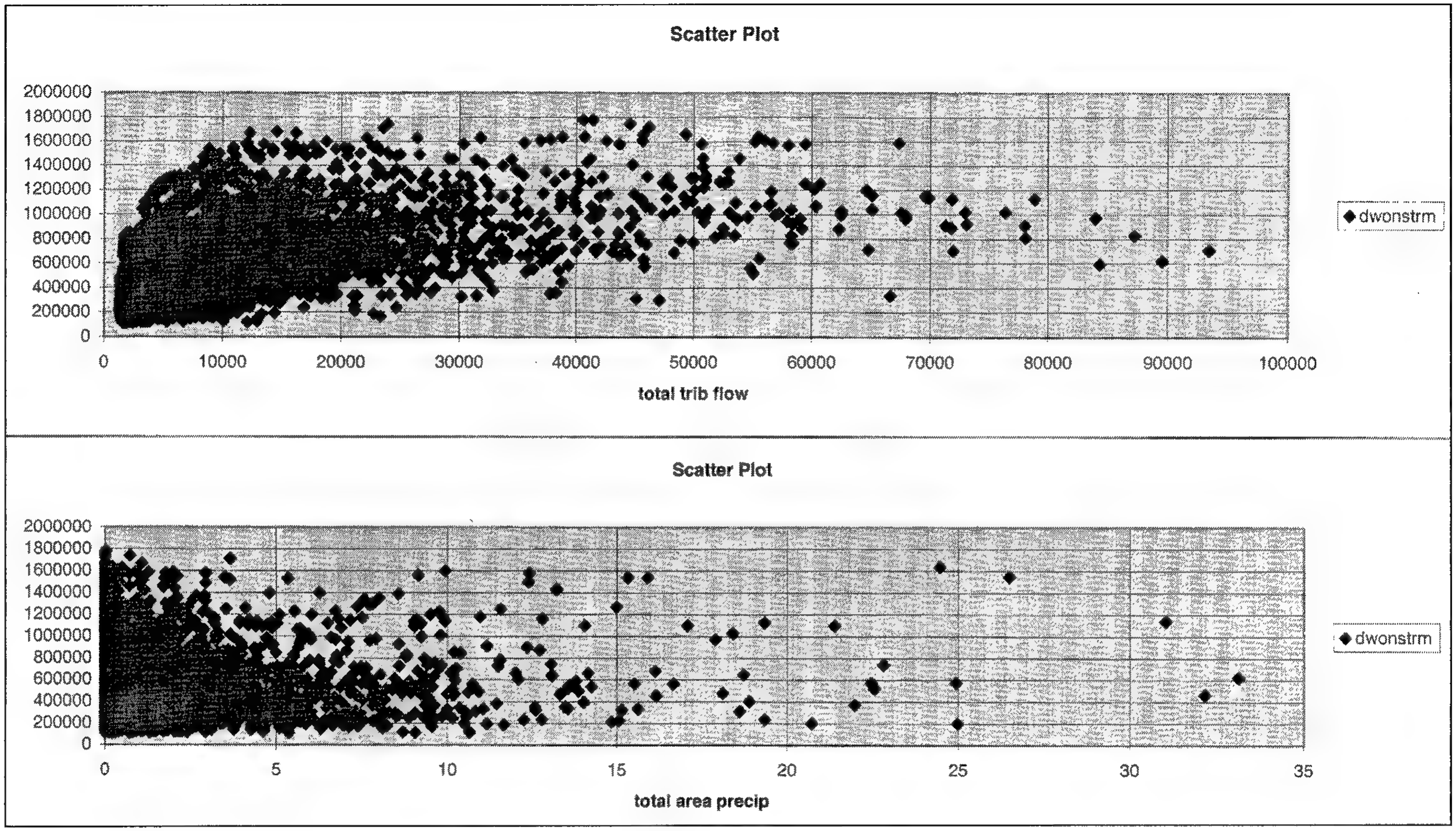


Figure 29c-d. Correlation of riverflow (cfs) between (c) total tributary (cfs) and (d) total precipitation (inches) and Memphis riverflow (y-axis)

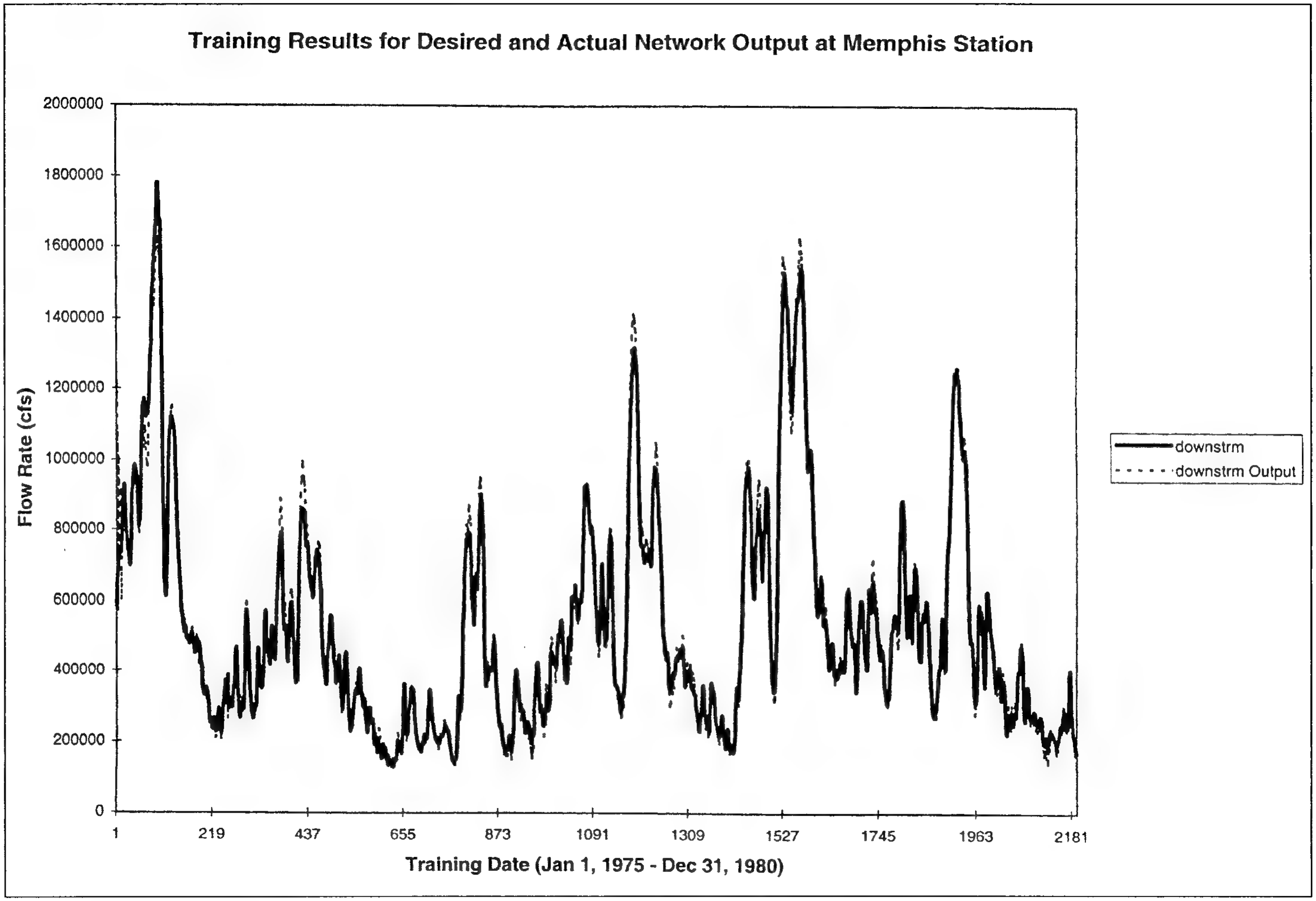


Figure 30. Training results for Mississippi River segment riverflow (cfs) prediction model with two upstream inputs (actual (dashed line) versus model (solid line))

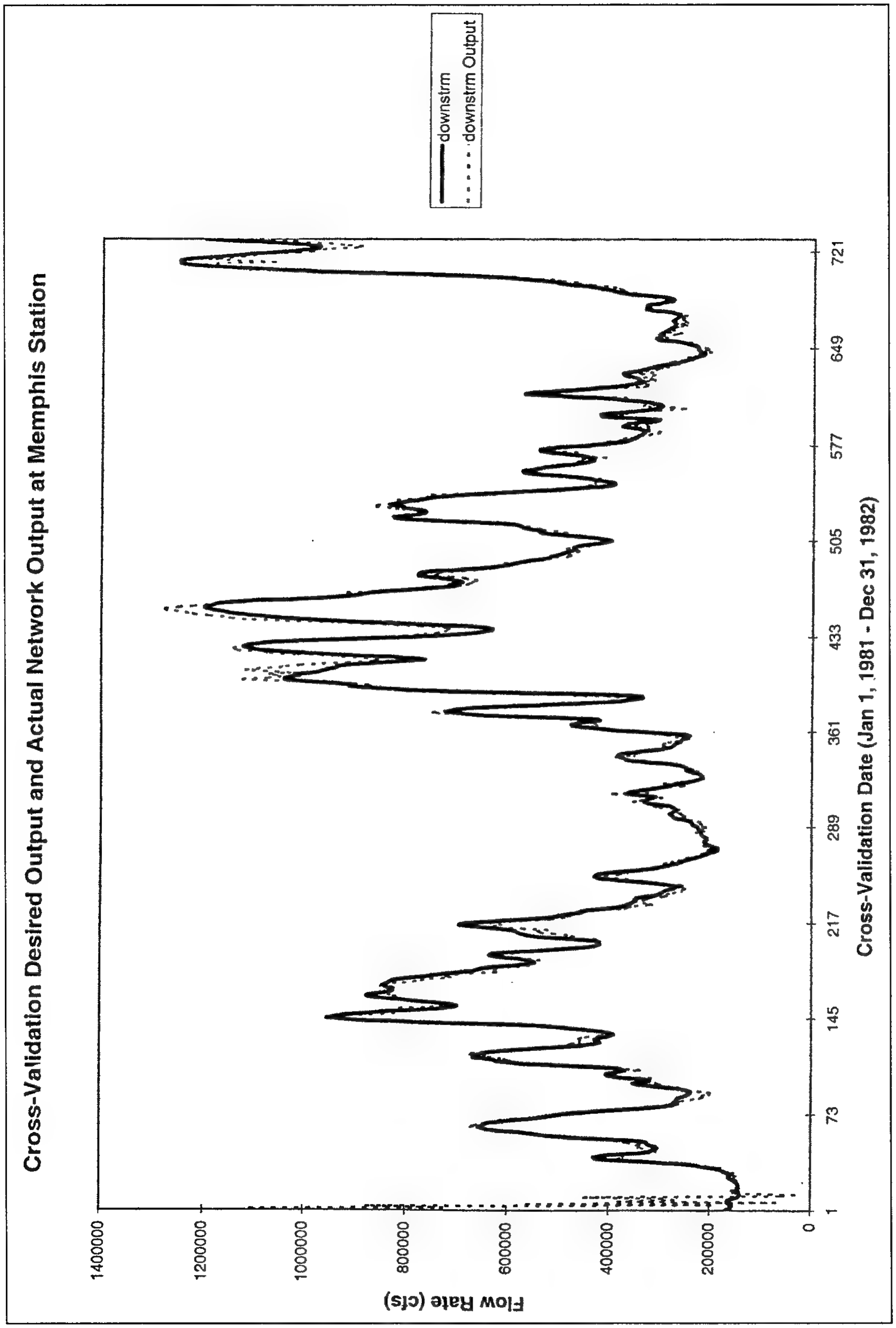


Figure 31. Cross-Validation results for Mississippi River segment riverflow (cfs) prediction model with two upstream inputs (actual (dashed line) versus model (solid line))

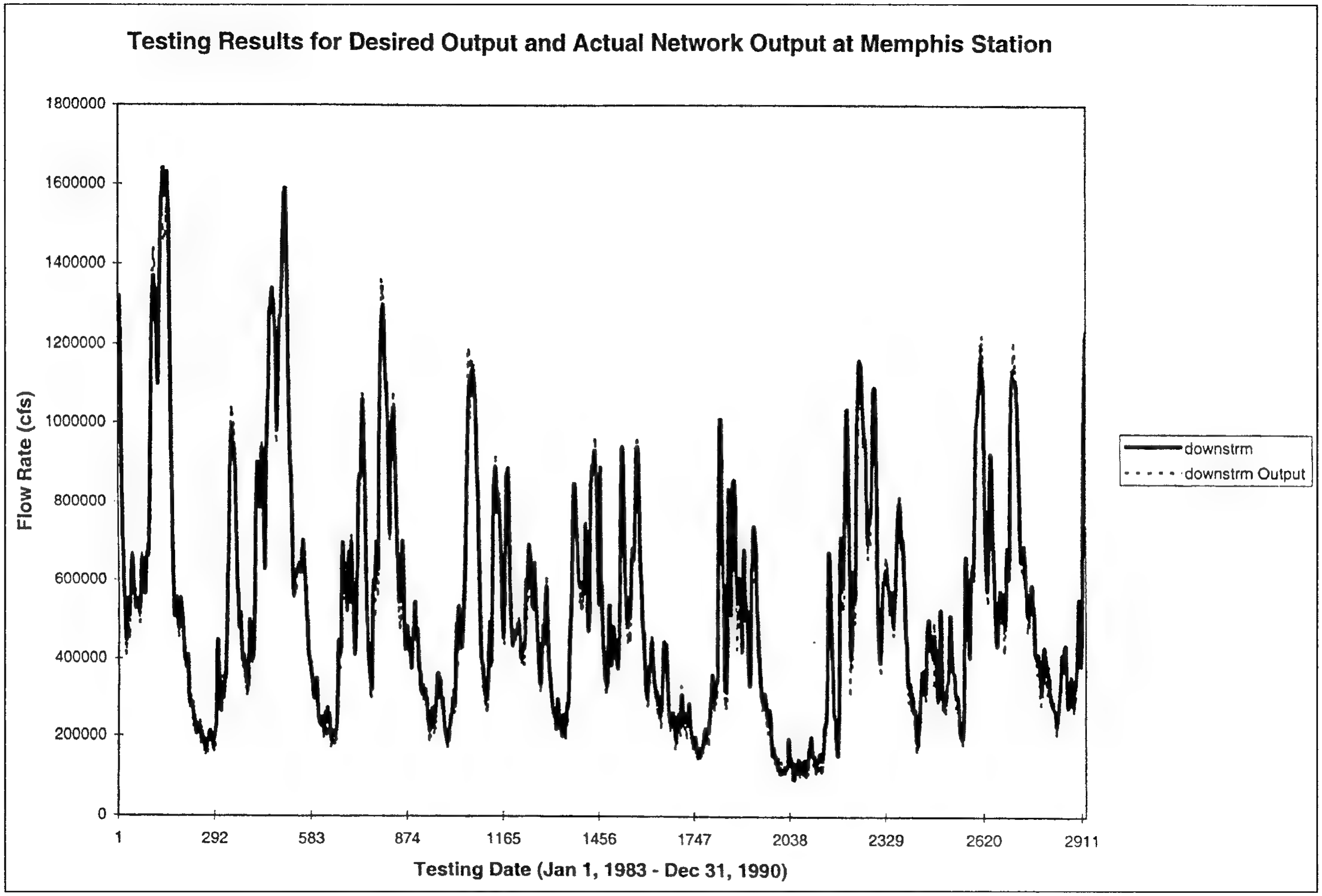


Figure 32. Testing results for Mississippi River segment riverflow (cfs) prediction model with two upstream inputs (actual (dashed line) versus model (solid line))

to SigmoidAxon as the output activation function and increasing the total iteration numbers (4000) for training, this model still shows the lowest performance among these combinations. This is sometimes the case with ANN modeling procedures. The more familiar the neural net modeler is with model design, the easier it becomes to determine the correct "recipe" in producing a reliable artificial neural network model.

Table 3 summarizes the performance for this variable input ANN build scenario. No additional improvement is gained from both the total rainfall and total tributary flow. The reason could be the hydrologic process is a highly nonlinear relationship. The rainfall contribution to the flow system is related to the infiltration rate, soil moisture content and other interactions. We concluded contribution of tributary flow is very minor compared to the magnitude of Mississippi River flow scale. The physical reason requires further research, such as using the difference between downstream and upstream flows as the model output to identify the contribution of total rainfall and total and/or individual tributary factors.

Table 3
Performance Analysis of Mississippi River Segment R - R Model

	Training		Cross-Validation		Testing	
Input × Output	**NMSE**	**r**	**NMSE**	**r**	**NMSE**	**r**
2 × 1	0.0190	0.9905	0.0774	0.9638	0.0210	0.9911
3 × 1 (Precipitation)	0.0211	0.9895	0.0956	0.9545	0.0226	0.9906
3 × 1 (Flow)	0.0216	0.9892	0.0974	0.9527	0.0260	0.9886
4 × 1	0.0503	0.9751	0.1384	0.9306	0.0552	0.9728

NMSE = Normalized Mean Squared Error.
r = Correlation Coefficient.

Another alternative to check the performance of network training, such as the sensitivity analysis, is to know the effect that each of the inputs is having on the network output. This provides feedback as to which input variables are the most significant. The process by removing the insignificant variables can reduce the size of the network, which in turn reduces the complexity and the training times. The basic idea of the sensitivity analysis about mean is that the inputs to the network are shifted slightly and the corresponding change in the output is reported either as a percentage or a raw difference. Figure 33 shows the significance for each input related to the output. Our result from this study indicated the total rainfall and total tributary flow still remain minor influences to the model.

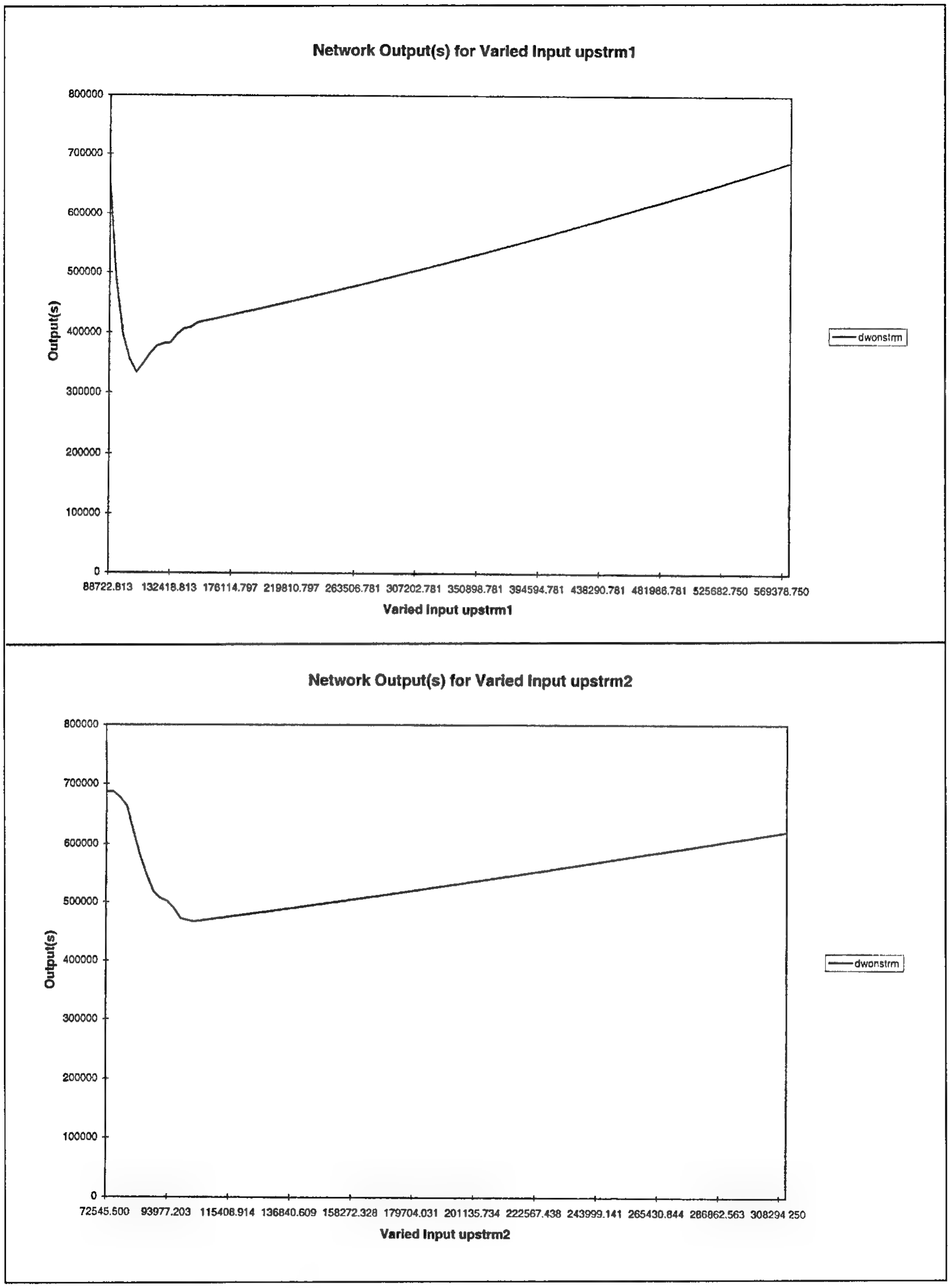
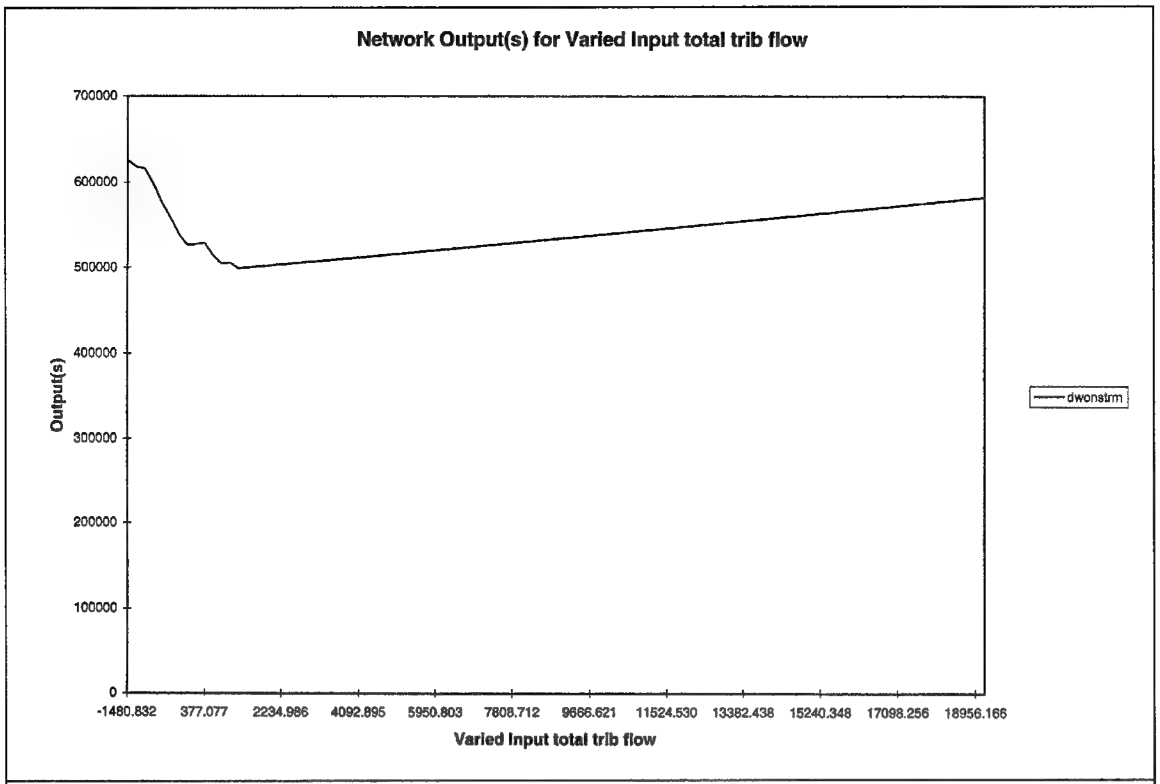


Figure 33a-b. The sensitivity analysis about mean for four-inputs/one output riverflow prediction model of Mississippi River segment (a) upstream station Thebes (b) upstream station Metropolis



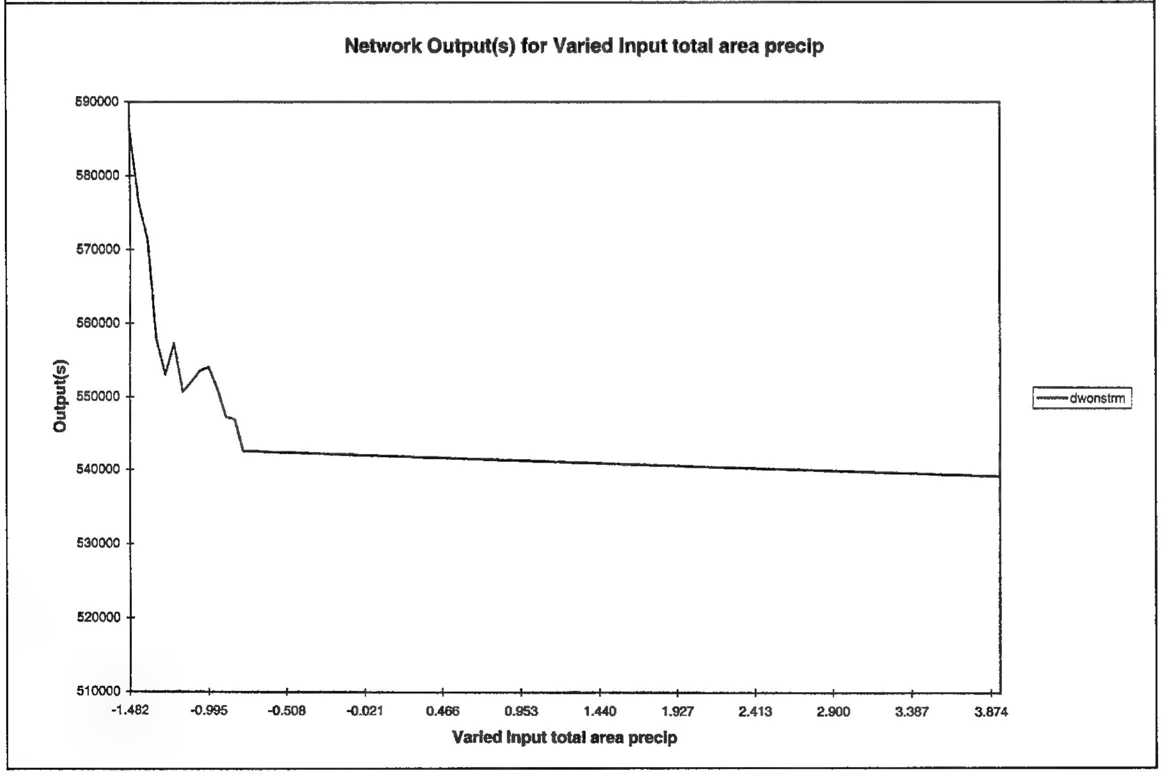


Figure 33c-d. The sensitivity analysis about mean for four-inputs/one output riverflow prediction model of Mississippi River segment (c) total tributary flow (d) total area precipitation

5 Discussions

To fully develop an ANN model requires the knowledge of existing methods necessary to address specific problem solving. This problem solving is usually in the form of choosing a mathematical algorithm, as applied to an artificial neural network, to determine the "best fit" of data that best represents a specified nonlinear data relationship, having the best global minimum results, in the shortest time. The course of constructing reliable models usually, from experience, requires several trial-and-error attempts at first. This chapter addresses several findings and particular goals investigated with our research using ANN modeling for a general hydrologic forecasting system and what was determined "best" depending on the specific situation.

Forecasting Reliability due to the Length of Training Record

It is interesting to know how reliable a prediction would be if only limited data were available. This was demonstrated in our Sava River Study by selecting a single station and repeating the model run with different record lengths. The river stage at the Zupanja site was selected to perform this test.

Nine test runs with MLPs were conducted with different lengths of training, cross-validation, and testing data with forecasting ranges from 1 day to 3 days. The results of testing are summarized in Table 4. Six statistical parameters were used to determine the prediction reliability. This table provides the forecasting reliability giving the length of record and expected criterion of accuracy for this station. The reliability usually decreased, as the training record length got shorter. For example, for only a 3 months' record, a 3-day prediction has over 1-m prediction error (1.02m) and the correlation coefficient (CC) is about 0.92.

Prediction Reliability due to Approach Algorithms

With computers continually advancing in technology and speed, the training time required to operate different algorithms in artificial neural networks may no longer be such a critical factor if the training record is short and the design architecture is not complex. The testing accuracy could worsen if the selection of the algorithm to represent the problem is not proper for the modeling simulation.

Table 4
Prediction Reliability Due to the Length of Training Record for Sava River Stage Prediction Model

		MSE	NMSE	MAE	Min Abs E	Max Abs Error	r
1 yr training	1 day p.	0.0821	0.0233	0.2015	0.0002	1.3868	0.9884
	2 day p.	0.2838	0.0808	0.4020	0.0027	2.3479	0.9595
	3 day p.	0.5554	0.1583	05802	0.0036	3.2460	0.9203
6 mo training	1 day p.	0.1125	0.0241	0.2336	0.0004	2.0161	0.9885
	2 day p.	0.3413	0.0731	0.4075	0.0001	3.3268	0.9628
	3 day p.	0.6491	0.1390	0.5990	0.0024	3.8329	0.9283
3 mo training	1 day p.	0.6955	0.2053	0.6783	0.0041	1.5347	0.9731
	2 day p.	0.9933	0.3041	0.8544	0.0034	1.7320	0.9489
	3 day p.	1.3350	0.4227	1.0194	0.0201	1.9215	0.9174

Four different algorithms for the example of the river flow prediction of the Mississippi River were used to demonstrate this comparison (Table 5). While the traditional BP algorithm without time shift input process showed the least accuracy, the Recurrent Algorithm represented the best results. The results indicate that the information from input to output mapping with certain memory length and strong nonlinearity can best describe this hydrological phenomenon. In this examination, the maximum iteration number (1,000) was set; therefore, the other algorithms may perform good training if they have more time to converge, such as with a Time-Lagged Algorithm (result being very close to the results with a Recurrent Network). The same test (Table 6) was also investigated by using the upstream-downstream river stage data set from the Sava River. While the TLRN derived the best result, the Radial Basis Function (RBF) algorithm obtained the largest error. The poor performance of testing for the RBF is mainly due to the high NMSE obtained during the cross-validation process.

Table 5
Prediction Reliability Due to Approach Algorithms for Mississippi River Segment Model

		Backpropagation	Backpropagation with Time Shift Input	Time-Delay	Recurrent
Training	NMSE	0.0966	0.0303	0.0194	0.0168
	R	0.9505	0.9847	0.9903	0.9918
Cross-V	NMSE	0.1448	0.0416	0.0665	0.0652
	R	0.9286	0.9807	0.9679	0.9680
Testing	NMSE	0.1042	0.0344	0.0210	0.0171
	R	0.9475	0.9834	0.9909	0.9922

Table 6
Comparison of Model Prediction Reliability Due to Approach Algorithms

		Backpropagation	Backpropagation with Time Shift Input	Time-Delay	Recurrent
Training	NMSE	0.0966	0.0303	0.0194	0.0168
	R	0.9505	0.9847	0.9903	0.9918
Cross-V	NMSE	0.1448	0.0416	0.0665	0.0652
	R	0.9286	0.9807	0.9679	0.9680
Testing	NMSE	0.1042	0.0344	0.0210	0.0171
	R	0.9475	0.9834	0.9909	0.9922

In summary, the Time-Lagged and Recurrent Algorithms are good candidates for hydrological forecasting; both in flow or stage data analysis. The performance of testing is also related to the learning processes such as learning rate and momentum factor. It requires further testing for other algorithms.

Testing Accuracy due to the order of Data Arrangement

The regular data arrangement for conducting the learning and reasoning processes is to use the time sequence order. For example, the training data set usually uses the earliest occurred information. However, for real application, such as missing data recovery, it might need to solve the interpolation or extrapolation problem. The accuracy of this prediction due to the order of training, cross-validation, and testing has to be examined.

An approach using an arrangement of four different data sequences was designed appearing as Figure 34 for the Mississippi River two-input/one output application. The RUN0 is the original arrangement (first 6 years' as training, the next two years as cross-validation, and the last eight years as the testing). The performance of the NMSE and correlation coefficient for each case is summarized as Table 7. The excellent performance and similar result were found for each case. However, the better correlation coefficient was obtained when using the latest portion of data set as the training information.

Table 7
Performance Analysis of ANN due to Data Arrangement

	Training		Cross-Validation		Testing	
	NMSE	r	NMSE	r	NMSE	r
Run 0	0.0190	0.9905	0.0744	0.9638	0.0210	0.9911
Run 1	0.0232	0.9883	0.0265	0.9895	0.0169	0.9925
Run 2	0.0159	0.9920	0.0389	0.9816	0.0259	0.9896
Run 3	0.0177	0.9911	0.0191	0.9905	0.0208	0.9915

NMSE = Normalized Mean Squared Error.
r = Correlation Coefficient.

Selection of Activation Functions for Time-Lagged Recurrent Networks

From the Sava and Mississippi Rivers studies, the TLRN was a very attractive and appropriate model for processing temporal information. Since the training algorithm used with TLRNs is more advanced than standard BP, the selection of these activation functions for the hidden layers and output layer is crucial to obtain good learning process. For the hidden layer process element, the TLRN has special memory system, such as TDNNAxon, GammaAxon, and LaguarreAxon. The TDNN memory structure is simply a cascade of ideal delays. The gamma memory is a cascade of leaky integration. The Laguarre memory is slightly more sophisticated than the gamma memory in that it orthogonalizes the memory space.

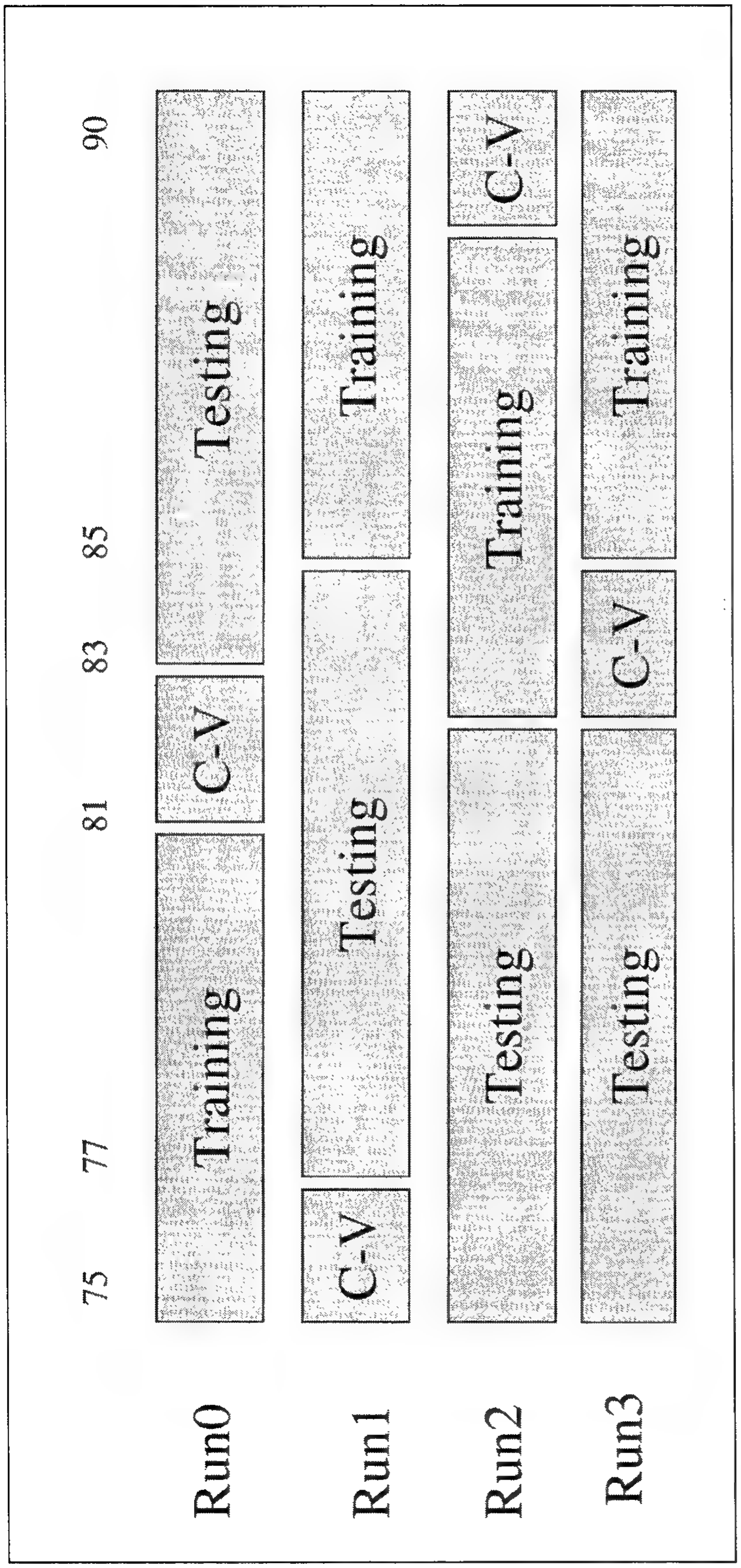


Figure 34. Testing network accuracy study for different data arrangement order for Mississippi River segment model

For the output layer process element, several activation functions, such as LinearAxon and SigmoidAxon, can be selected. This section (Table 8) summarizes the test for different combinations of activation functions of hidden layer and output layer using forty years monthly mean flow data at Sava River (three input/one output system).

Table 8
Performance Analysis of Sava River, 3 Input × 1 Output, 40 yr Monthly Data

Transfer Function	Training		Cross-Validation		Correlation Coefficient (r) Comparison			
Hidden Layer/Output Layer	FMSE	MMSE	FMSE	MMSE	Time (sec)	Training	C - V	Testing
TDNN/Linear Axon	0.0030	0.0030	0.0024	0.0024	10	0.9829	0.9892	0.9898
Gamma/Linear Axon	0.0029	0.0029	0.0018	0.0018	8	0.9835	0.9823	0.9934
Laguarre/Linear Axon	0.0029	0.0029	0.0018	0.0018	20	0.9836	0.9922	0.9932
TDNN/Tanh Axon	0.0041	0.0041	0.0091	0.0087	13	0.9712	0.9625	0.945
TDNN/Sigmoid Axon	0.0128	0.0128	0.0184	0.0184	11	0.7565	0.7494	0.7954
TDNN/Linear Tanh Axon	0.0067	0.0067	0.0254	0.0249	10	0.9541	0.8706	0.9092
TDNN/Bias Axon	0.0034	0.0034	0.0036	0.0036	10	0.9861	0.9906	0.9915
TDNN/Axon	0.0023	0.0023	0.0026	0.0026	9	0.9865	0.9888	0.9906
TDNN/Linear Sigmoid Axon	0.0105	0.0105	0.0153	0.0153	10	0.7243	0.6963	0.7531

FMSE = Final Mean Squared Error.
MMSE = Minimum Mean Squared Error.
r = Correlation Coefficient.

The best combinations are TDNN/Linear Axon, Gamma/Linear Axon, Laguarre/LinearAxon, TDNN/Bias Axon, and TDNN/Axon. The performance analysis, using the correlation coefficient as comparison, was over 0.99 for each of these combinations. However, the Laguarre/Linear Axon takes more time for train. The Sigmoid Axon and Linear Sigmoid Axon are slow convergent activation functions. The LinearAxon is a good choice for selection as an output process element and had the best performance overall. The Tanh (Hyperbolic Tangent) Axon family tended to overtrain, however.

Testing Accuracy Due to Data Representation for the Mississippi River Segment Model

In the previous chapter, it was concluded that an accurate downstream flow prediction while using ANNs, was a result of the gage data input from the two upstream flows. The total tributary flow and regional rainfall input contributed very little to improve the accuracy of the learning process for a time-lagged algorithm in ANNs. A sensitivity analysis or principal component analysis can be used to identify the most significant input variables to the desired output. However, in order to investigate how the data representation affects the ANNs model performance, a model run from a series of input/output topology was performed. In addition, another algorithm, Jordan-Elman partial recurrent network, was also considered to compare the performance of time-lagged networks.

Two parameter structures (Columns 1 and 2 of Table 9), namely an input/output parameter and an input variable parameter, were identified. The first parameter represents the number of input and output variables. For example, the parameter 4 × 1 represents a four-input and one-output system. The second parameter involves three variables that address the input data: upstream flow gages, tributary flow and precipitation.

Table 9
Testing Accuracy Due to Data Representation for the Mississippi River Segment Model

Input/Output	Upstream Flow, Tributary, Rainfall	Jordan-Elman			Time Lagged Recurrent		
			NMSE	r		NMSE	r
2 × 1	(2, 0, 0)	o	0.0227	0.9908	o	0.0210	0.9911
4 × 1	(2, 2, 0)	o	0.0206	0.9917	*	0.2267	0.9545
3 × 1	(2, 1, 0)	o	0.0231	0.9907	o	0.0260	0.9886
6 × 1	(2, 4, 0)	o	0.0269	0.9896	*	0.1152	0.9606
8 × 1	(2, 4, 2)	o	0.0251	0.9898	*	0.1016	0.9614
10 × 1	(2, 4, 4)	*	0.8165	0.9364	*	0.1138	0.9497
4 × 1	(2, 1, 1)	o	0.0214	0.9912	*	0.0552	0.9728
16 × 1	(2, 4, 10)	*	0.2425	0.8837	*	0.0963	0.9528

Transfer function combinations (hidden layer/output layer)

o = Linear Axon/Linear Axon
* = Sigmoid Axon/Sigmoid Axon
NMSE = Normalized Mean Squared Error
r = Correlation Coefficient

Two conditions were considered to represent tributary and rainfall information. For simplification in training the ANN model, the total tributary and total rainfall data were obtained by the summation of four individual tributaries and sixteen rainfall stations respectively. However, this assumption led to an inconsistency with the basic principles of a hydrological process–those nonlinearity and time-delay affects involved when constructing a model. Which meant, under these conditions, each input variable had the same travel time to pass the signal to the output variable. This summation process assumed that a linear superposition principle applied, thus leading to a poor performance by the ANN model. Referring to Table 9, the second column lists the number of inputs in parenthesis based on those input values chosen, incorporating either upstream flow, tributary and rainfall data or a combination of two or all three values for the modeling process.

From the preliminary data analysis, it was found that the Obion and Loosahatchie Rivers and four rainfall stations correlated to the downstream flow. Therefore, the value 2 individually represents these two rivers and the value 4 individually represents all four rivers as the second element (tributary) of the input variable parameter. Two of the four rainfall stations, one station near the confluence of the main stem of Mississippi and Ohio Rivers and the other station close to Memphis, were slightly higher in correlation than the other two stations. Similarly, the value 2 of the third element for input variable parameter represents these two indicated two precipitation stations. The value 4 of that element represents four correlated precipitation stations.

Two pairs of nonlinear transfer functions (activation), namely Linear Axon/Linear Axon and Sigmoid Axon/Sigmoid Axon were used to describe the nonlinearity of corresponding hidden/output layers for both algorithms. The maximum iteration was 1,000 and the NMSE and correlation coefficients were used to represent the performance for testing.

Several findings were summarized in Table 9:

a. The Jordan-Elman partial recurrent network, in general, had better performance than the Time-lagged network, particularly when the number of input variables increases. This implied that the Jordan-Elman network has a stronger tolerance for "noise" in a system.

b. Rows 1, 2, 4, 5, 6, and 8 show the increasing dimension of individual tributary and rainfall information. From both NMSE and correlation coefficient, it can be concluded that the optimal performance occurred with an input variable parameter selection of (2,2,2).

c. Rows 3 and 4 show the data representing tributary flow. Although there was no significant difference of performance between these two cases for Jordan-Elman networks, both networks' performance worsened with an input variable parameter (2,4,0). This indicated a bad performance would be expected if more uncorrected inputs were included.

d. Rows 7 and 8 show the combined influence of both tributary and rainfall for data representation. Excellent performance was obtained by the input variable parameter (2,1,1) for a Jordan-Elman network. This indicated a simplification of data representation could contribute the improvement of performance. However, it might not be used to predict local simulation behavior since the local effect has been transferred to a summation variable.

6 Conclusions

ANN algorithms were successfully applied to two different scale watershed systems for river flow and stage prediction, addressing two primary hydrological phenomenons: time delay and nonlinearity.

In the lower portions of the Mississippi River, river flow characteristics at Memphis, TN can be predicted with a high degree of accuracy from two upstream gages, even with no rainfall data and tributary flow data provided. This model also can be used to analyze the influence (if any) by the flow input of the Ohio River downstream on the Mississippi River. The study shows that tributary flow contributes to the river flow prediction in the downstream portions of the river, but very little additional prediction accuracy was gained. Some possible explanations for this: With the compilation of tributary flow into one value from the summation of four separate data values, along with the summation of ten separate precipitation collection stations values into one value representing total rainfall, we are assuming that these non-linear relationships are and should be treated like a linear process. In essence, we are combining unlikely relationships between linear processes and group time-delay phenomena, and treating as a neural network that can be solved using a simple time delay problem. With additional input variables, performance will be affected by additional noise created by the network. It can be concluded that by using the minimum input variables that have good correlation to existing output variables, optimal performance will be achieved.

Less accurate results were obtained for the Sava River daily flow study mainly due to the limited length of available data sets. The excellent performance was found by ANN model for forty years monthly mean data set for the Sava River. With two upstream data sets available, the model can accurately predict the downstream monthly flow.

The prediction for river stage/flow can be obtained by generating the relationship between training length and performance parameters. The proper selection for a solution algorithm and activations for processing nonlinearity could help increase the model accuracy. The data arrangement sequence for learning and reasoning processes has very little influence to the model accuracy. However, using the last portion of data as the training set might get better performance when testing. The time-lagged recurrent network and Jordan-Elman partial recurrent network are good selections for investigating the river flow/stage forecasting system. TDNN and LinearAxon is a candidate for process element of hidden layer and output layer respectively.

In conclusion, the best performance of an ANN for flow prediction, as all the ANNs modeling study, heavily depends on not only the length of the data sets but also whether the most significant patterns were included in the process.

References

Carriere, P.S. Mohaghegh, and R. Gaskari, 1996, "Performance of a Virtual Runoff Hydrographic System," *Journal of Water Resources Planning and Management*, Vol. 122, No 6. 120-125.

Fernando, D.A.K. and A. W. Jayawardena, 1998, "Runoff Forecasting Using RBF Networks with OLS Algorithm," *Journal of Hydrologic Engineering*, 3(3) 203-209.

Hjelmfelt, A. T. and M. Wang, 1996, "Predicting Runoff Using Artificial Neural Networks," *Surface Water Hydrology*, 233-244.

Hsu K, H. V. Gupta, and S. Sorooshian, 1998, " Streamflow Forecasting Using Artificial Neural Networks," *ASCE Water Resources Engineering Conference '98*, 967-972.

Karunanithi, N., W. J. Grenney, D. Whiteley, and K. Bovee, 1994, "Neural Networks for River Flow Prediction," *ASCE Journal of Computing Civil Engineering*, 8(2), 201-220.

Lippmann, R. P., 1987, "An Introduction to Computing with Neural Nets", *IEEE ASSP Magazine*, April 1987, 4-22.

NeuroSolutions, 1998, "The Neural Network Simulation Environment", Getting Started Manual, NeuroDimensions, Inc., Gainsville, Florida.

Rogers, S. K., K.L. Priddy, P.E. Keller, L. J. Kangas, 1998, "Cognitive Systems at Battelle: Artificial Neural Networks", Webpage: www.battelle.org/cogsys/ann.htm.

Rumelhart, D. E., G. E. Hinton, R. J. Williams, September 1985, *Learning Internal Representations by Error Propagation*, Institute for Cognitive Science, Report Number 8506, 34 p.

Skapura, D. M., 1996, *Building Neural Networks*, ACM Press, New York, 286 p.

Wasserman, P. D., 1989, *Neural Computing Theory and Practice*. Van Nostrand Reinhold, New York, 230 p.

Zhang B., and R. S. Govindaraju, 1998, "Using Modular Neural Networks to Predict Watershed Runoff," *ASCE Water Resources Engineering Conference '98*, 897-902.

Zhang B. 1998. "Geomorphological Artificial Neural Networks for Predicting Watershed Response to Precipitation Events," Dissertation Proposal, Purdue University, West Lafayette, IN.

Appendix A
Fundamental Aspects of ANNs

Learning Strategies

Neural network models use supervised and unsupervised learning modes (Figure A1). Learning entails training the network by presenting training patterns to its visible layers. The aim of learning is to set the connection weights and internal representation so that the desired output is obtained. The most common performance measure of learning is to compare the mean squared error between the network output and the actual output value.

In supervised learning, the neural network is trained with a data set in the form of input-output data pairs, provided by a "teacher". The network receives the input values and calculates an output, which is then compared with the correct value. The aim of the training is to teach the network to map a correct output vector for every input vector by developing appropriate connections in the model. A parameter searching procedure aims to minimize the error function and to obtain the optimal weights. After reaching the point where there are no additional weight changes required, that the neural network reaches global stability and is "trained."

Unsupervised learning is conducted without the teacher, using a training data set, which consists only of input data. The neural network is training itself to achieve stability, when the weights attain constant values. This type of learning is suitable for applications dealing with the clustering of input data, reduction of input dimensions, data compression and similar types of applications. Networks trained by this method are called self-organizing networks. Examples of this: Adaptive Resonance Theory (ART) Network and Kohonen's Self-Organizing Feature Maps (SOFM) (Hush, Horne 1993). The network can organize the inputs in any way it wishes. The processing elements can be organized in clusters with either competition or cooperation between the clusters occurring.

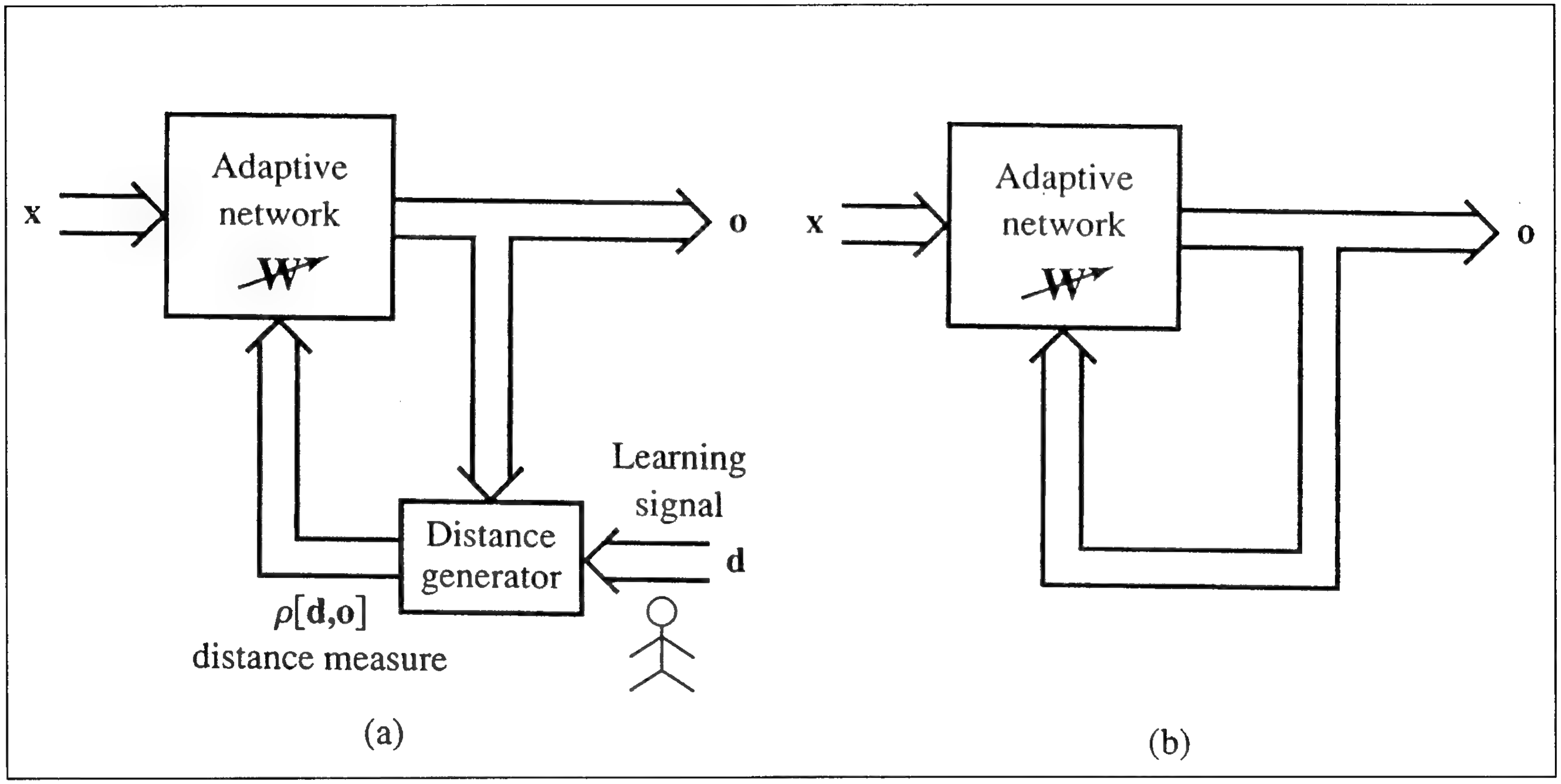


Figure A1. (a) Supervised learning and (b) unsupervised learning models

Major Components for Building Artificial Neural Networks

(1) Training. A network learns by adjusting the biases and weights that link its neurons. Before actually training begins, a network's weights and biases must be set to small random values. Obtaining the best set of weights that will be utilized during the cross-validation and testing process is the main goal here.

(2) Cross-validation. Method used in conjunction with Step 1, it is a process to monitor and finalize established weights derived through the training process. This is necessary to prevent over training. Over training can actually degrade performance on the test set. A portion of the training set should be set-aside for the purpose of Cross-Validation.

(3) Testing. Proving the performance generalization of the neural networks and establishing the best set of weights to use to derive a global minimum error result.

Procedures Used to Construct an ANN Model

The neural network design and use life cycle is a complex dynamic process with many steps. NeuroSolution, Inc. (1999) summarizes the following steps used to construct a neural network:

(1) Understand the data. Neural networks cannot be used as "black boxes", even in the best circumstances. There is no substitute for a firm understanding of the data. Explore the data in as many ways as possible. First: Try to understand the physical process that produced the data.

(2) Plot the data. Examine the statistics for interpreting the data. Finally, Use digital signal processing analysis techniques to understand the data in the frequency domain.
Preprocessing the data: Taking the insight gained from "understanding the data" and encoding it into the data.

(3) Choose a desired Input-Output Mapping. Decide what the neural network is to accomplish. In particular, what is to be the desired input-output relationship? Sometimes this can require laborious hand coding of the data.

(4) Choose a Neural Architecture. For regression, always start out with a linear network. For classification, always start out with a linear discriminant classifier. Even if these networks do not perform well, they provide a baseline comparison for other networks as you graduate in complexity. Also, a consideration here is whether an unsupervised network can perform the desired input-output mapping.

(5) Train the Network. If possible, monitor the training with a subset of the training exemplars set aside as a cross-validation set. If the data are too small to

use cross-validation, then stop the training when the learning curve first starts to level off.

(6) Repeat the Training. There is a high degree of variability in the performance of a network trained multiple times, but starting from different initial conditions. Therefore, the training should be repeated several times, varying the size of the network, and/or the learning parameters. Among those networks that perform the best (on the cross-validation set, if available), choose the one with the smallest number of free weights.

(7) Perform Sensitivity Analysis. Sensitivity analysis measures the effect of small changes in the input channels on the output, and is computed over the whole training set. It can be used to identify superfluous input channels. Eliminate those channels and repeat the training process.

(8) Test the Network on the New Data. This is where you put the network to use. If you have carefully followed the previous steps, the network should generalize well to new data.

(9) Update the Training. Occasionally, when enough data are accumulated, include old test data in with the existing training set, and repeat the entire training process.

Appendix B ANN Algorithms for Hydrologic Forecasting

Basic ANN Algorithms for Hydrologic Forecasting Systems

(1) Multi-layered Feed Forward Neural Network (same as Figure B1). Also known as Multilayer Perceptrons (MLPs), these networks are typically trained with static backpropagation. These networks have found their way into countless applications requiring static pattern classification. Their main advantage is that they are easy to use, straightforward in conceptual design, and that they can approximate any input/output map. The key disadvantages are that they train slowly, and require large amounts of training data (typically three times more training samples than network weights). (NeuroDimension, 1999)

(2) Time-Delayed Neural Network (TDNN), (Figure B1). This incorporates the use of a static network to process time series data by simply converting the temporal sequence into a static pattern by unfolding the sequence over time. That is, time is treated as another dimension in the problem. The process is accomplished by feeding the input sequence into a tapped delay line of finite extent, then feeding the taps from the delay line into a static neural network architecture like a MLP. TDNNs have been successfully used and applied to nonlinear time series prediction problems. (Hush, et al, 1993)

(3) Recurrent Neural Network (Figure B2). Consisting of two types: Fully recurrent networks feed back the hidden layer to itself. Partially recurrent networks start with a fully recurrent net and add a feedforward connection that bypasses the recurrence, effectively treating the recurrent part as a state memory. Recurrent networks have an infinite memory depth and thus find relationships through time as well as through the instantaneous input space. Most real-world data contains information in its time structure. These networks are state-of-the-art in nonlinear time series prediction, system identification, and temporal pattern classification. (NeuroDimension, 1999)

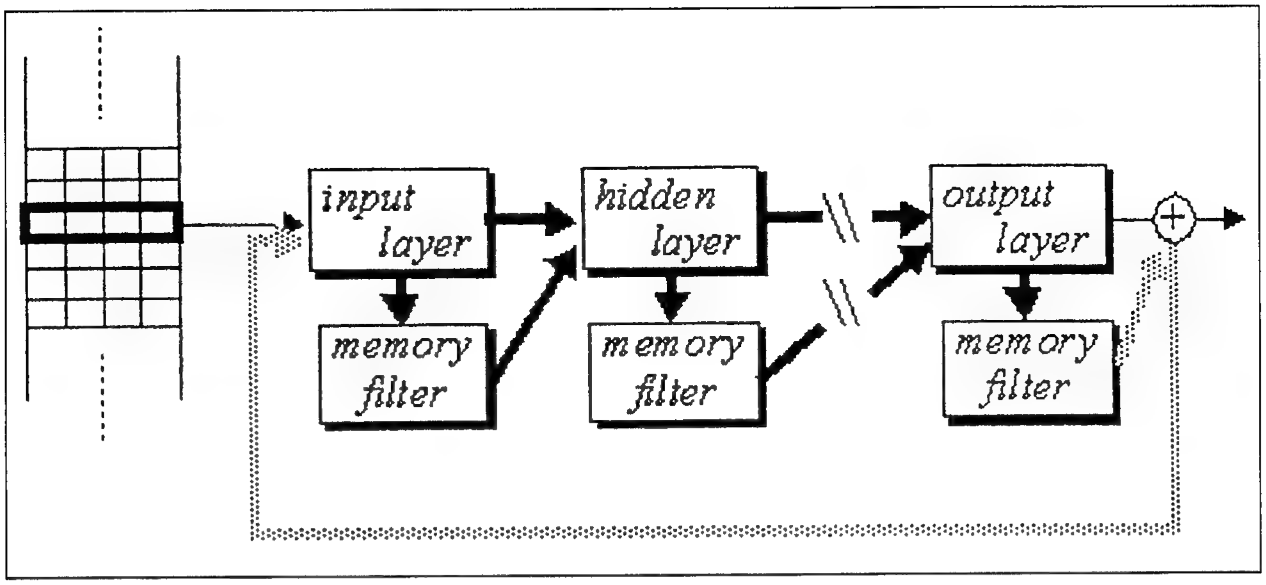


Figure B1. Time-lagged neural network

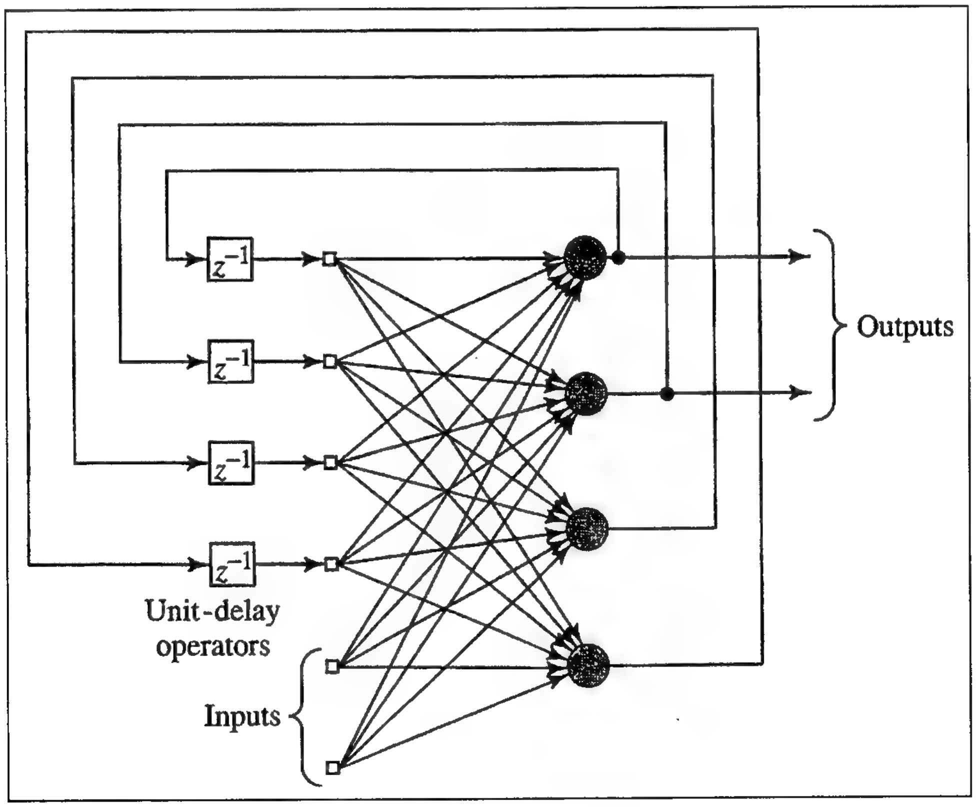


Figure B2. Recurrent neural network

(4) Self-Organizing Linear Output (SOLO), (Figure B3). Consists of a hybrid model structure that links a *s*elf-*o*rganizing feature map (SOFM) with a piece-wise locally *l*inear *o*utput mapping (LO). SOLO uses simple piecewise mappings to represent the system local input-output behavior. The overall structure results in a two-stage training procedure that is significantly less costly and easier to implement because it does not involve non-linear global optimization. (Hsu, et al, 1998)

(5) Radial-Basis Function Neural Network (Figure B4). A Radial Basis Function (RBF) network is a two-layer network whose output nodes form a linear combination of the basis (or kernel) functions computed by the hidden layer nodes. The basis functions in the hidden layer produce a localized response to input stimulus. That is, they produce a significant nonzero response only when the input falls within a localized region of the input space. For this reason the network is sometimes referred to as the "localized receptive field" network. The RBF network can be used for both classification and functional approximation, just like the MLP. In theory, the RBF network, like the MLP, is capable of forming an arbitrarily close approximation to any continuous nonlinear mapping. The primary difference between the two is the nature of their basis functions. The basis functions in the RBF network cover only small-localized regions. (Hush, et al, 1993)

Back-Propagation (BP) Training Algorithm

The basic architecture of an ANNs is the MLP. MLPs are feed-forward nets with one or more layers of nodes between the input and output nodes. These additional layers contain hidden units or nodes that are not directly connected to both the input and output nodes. MLPs overcome many of the limitations of single-layer perceptrons, but were generally not used in the past because effective training algorithms were not available. The development of new algorithms has changed this outlook significantly. The capabilities of MLPs stem from the nonlinearities used within nodes.

The BP Training Algorithm, which fully incorporates the MLP architecture, is currently the most general-purpose, commonly used neural-network paradigm. It is the most commonly used supervised training algorithm among the MLP. The development of the Back-Propagation Algorithm introduced a new method of modifying the network weights by minimizing the error between a target and computed objects. In back-propagation networks, the information is processed in the forward direction from the input layer to hidden layer(s) and then to the output layer.

Optimization of training weights

The objective of a Back-Propagation Network (BPN) is to find the weights that approximate target values of output with a selected accuracy. The Least-Mean-Square-Error Method along with the Generalized-Delta Rule is used to

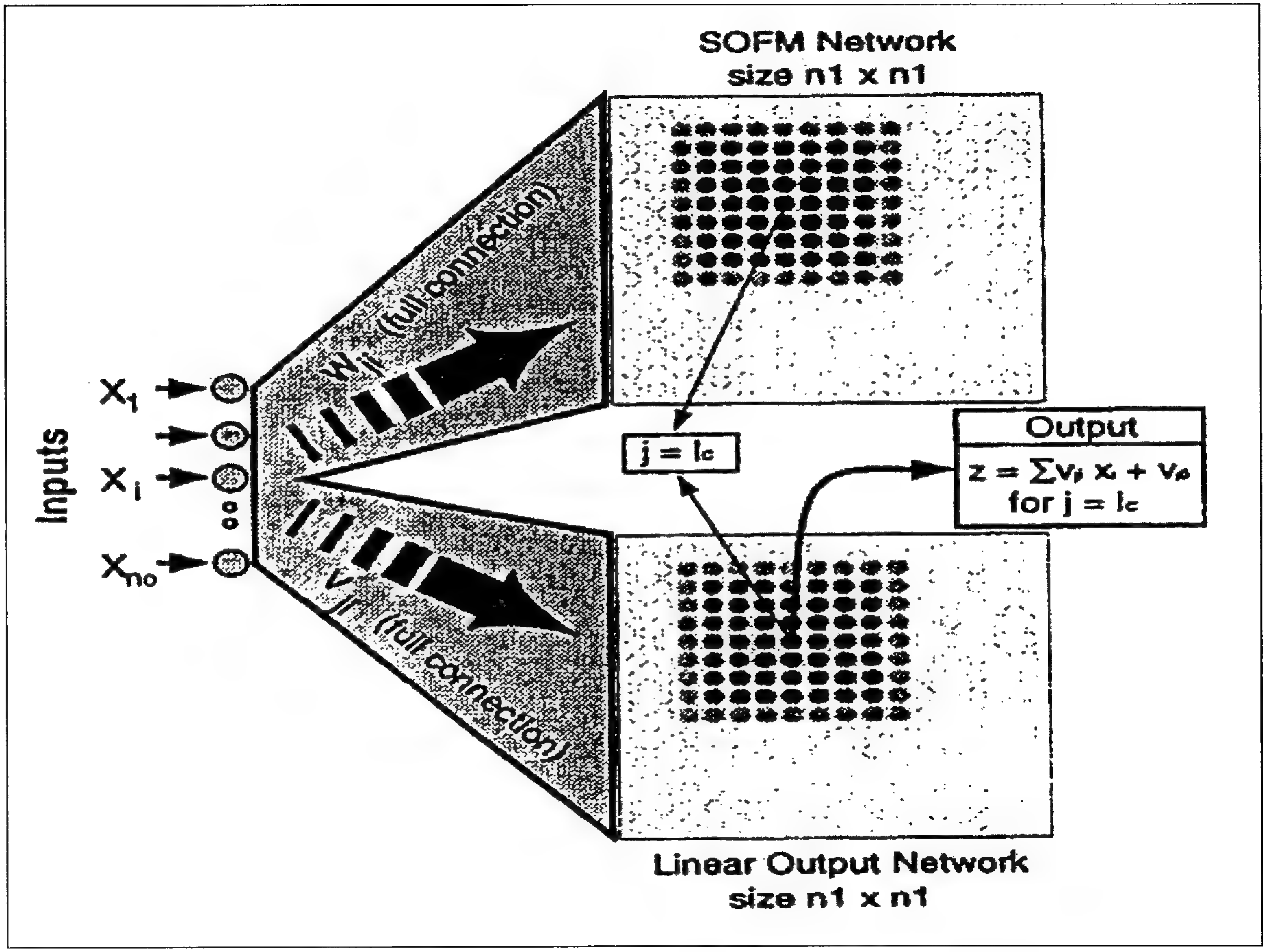


Figure B3. Self-organizing linear output network (SOLO)

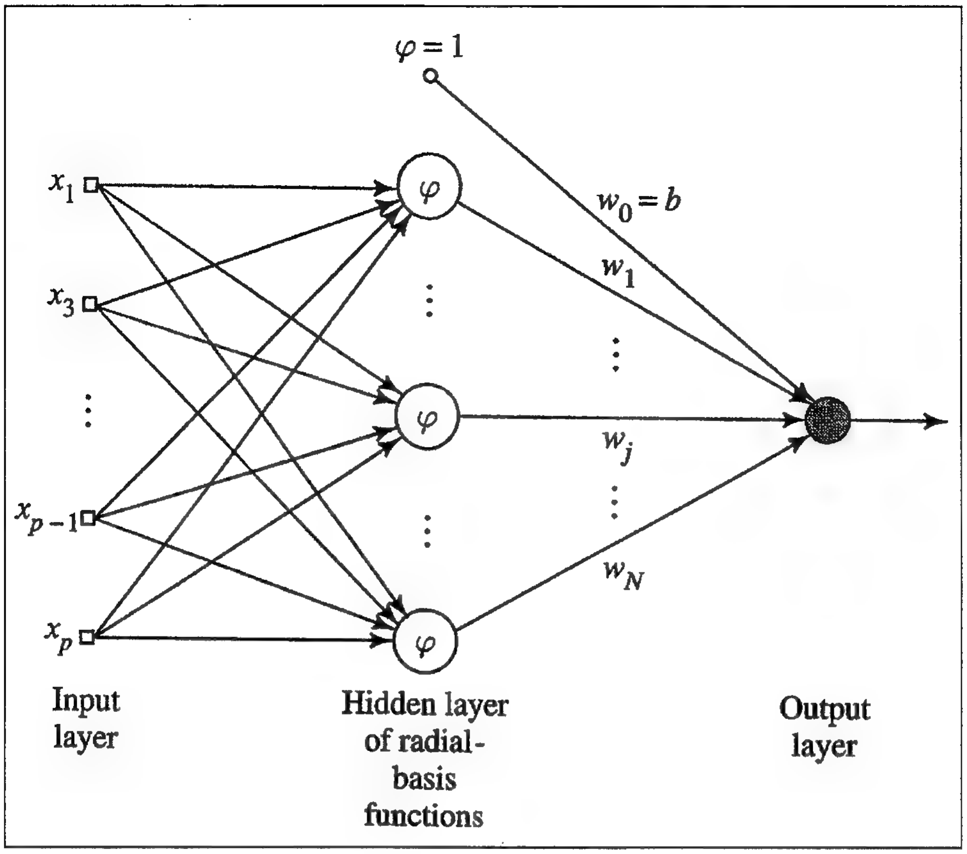


Figure B4. Radial-basis function neural network

optimize the network weights in BPN. The Gradient-Descent Method, along with the Chain Rule of Derivatives, is employed to modify the network weights.

Basically, the BP Algorithm attempts to develop a function that correctly represents the given data set. The adjustment of the interconnection weights during training employs a method known as "error back propagation" in which the weight associated with each connection is adjusted by an amount proportional to the strength of the signal in the connection and the total measure of the error. (Rummelhart, 1986)

When discussing the modification of network weights, the gradient descent method is analogous to an error-minimization process. Error-minimization is an attempt to fit a closed-form solution to a set of empirical data points, such that the solution deviates from the exact value by a minimal amount. In general, the back propagation-training algorithm is an iterative gradient algorithm designed to minimize the mean square error between the actual output of a multi-layer feed-forward perceptron and the desired output. It employs the techniques of propagating error terms required to adapt weights back from nodes in the output layer to nodes in lower (hidden) layers.

Operation of the BP algorithm

The overall operation of the BP Algorithm is as follows: The net is trained by initially selecting small random weights and internal thresholds and then presenting all training data repeatedly (Lippmann, 1987). The learning process begins with the presentation of an input pattern to the BPN. The input pattern is propagated through the entire network until an output pattern is produced. The BPN then makes use of the "generalized delta rule" to determine the error for the current pattern contributed by every unit in the network. Finally, each unit modifies its input connection weights slightly in a direction that reduces its error signal, and the process is repeated for the next pattern (Lippmann, 1987; Skapura, 1995).

The error between the output of the network and target outputs is computed at the end of each forward pass. If an error is higher than a selected value, the procedure continues with a reverse pass; otherwise training is stopped. In the reverse pass, or actual "back propagation" function of this algorithm, the weights in the network are modified using the error value. The modification of weights in the output layer is different from the modification of weights in the intermediate layers because target values do not exist. Therefore, BP uses the derivatives of the objective function with respect to the weights in the entire network to distribute the error to neurons in each layer in the entire network (Wasserman, 1990).

The next input/output set is applied and the connection weights are readjusted to minimize this new error. In this way, the BP algorithm can be seen to be a form of gradient descent for finding the minimum value of the multi-dimensional error function. This procedure is repeated until all training data sets have been applied. The whole process is repeated starting from the first data set again, once more, and continued until the total error for all data sets is sufficiently small and subsequent adjustments to the weights are inconsequential. The ANNs is now said to have learned a relationship between the input and output training data and a function that best describes this non-linear data set has been derived. Now, the neural network is "trained".

Mathematical description of BP

MLPs extend the perceptron with hidden layers, i.e., layers of processing elements that are not connected to the external world. There are two important characteristics of the MLP. First, its processing elements are nonlinear. The nonlinearity function must be smooth. Second, they are fully interconnected such that any element of a given layer feeds the entire next layer.

Zhang (1998) summarizes the procedures of the BP algorithm for three layer feed forward networks. The major steps are described as follows:

Each input neuron k receives input signal x_k ($k = 1, 2 \ldots m$, m is number of inputs) and sends this signal to all units in the following layers toward the output layer. Meanwhile, each hidden neuron i $(i=1,2...p)$ computes its total weighted input and applies to its activation function.

$$u_i(n) = \Sigma w_{ik}(n)\, x_k(n) \tag{1}$$

$$y_i = f(u_i(n)) \tag{2}$$

where w_{ik} is connection weight including bias. y_i is output of hidden neuron i and input to output layer.

Each output neuron j ($j=1,2... o$, o is number of output neurons) computes its total weighted input and applies to its activation function.

$$u_i(n) = \Sigma w_{ji}(n)\, y_i(n) \tag{3}$$

$$y_j(n) = f(u_j(n)) \tag{4}$$

Each output neuron j receives a target value corresponding to the input training pattern to evaluate the error term and computes its local gradient $\delta_j(n)$

$$\delta_j(n) = -e_j(n)\, f'(u_j)) \tag{5}$$

Then calculates and adjusts its connection weight

$$w_{ji}(n) = w_{ji}(n-1) + \eta\delta_j(n)\, y_i(n) + \alpha\Delta w_{ji}(n-1) \tag{6}$$

Each hidden neuron i receives propagated error term from output layer and computes its local gradient $\delta_I(n)$

$$\delta_I(n) = f'(u_i(n))\, \Sigma\, \delta_j(n)\, w_{ji} \tag{7}$$

then calculates and adjusts its connection weight

$$w_{ik}(n) = w_{ik}(n-1) + \eta\delta_i(n)\, x_k(n) + \alpha\Delta w_{ik}(n-1) \tag{8}$$

The most important parameters are η (learning rate) and α (momentum term). Momentum learning is an improvement to the straight gradient descent in the sense that a memory term (the past increment to the weight) is utilized to speed up and stabilize convergence.

Appendix C
Practical Implementation to an ANN Model

In this section, an overview for implementing an ANN model using the software, NeuroSolutions by NeuroDimensions, Inc., is discussed. A process to select comprehensive and user-friendly software has been conducted. For the supervised learning with our prediction capability, NeuroSolutions, a premier neural network simulation environment software, is adopted. We chose the "Professional" Level of this software package that was easily loaded on a standard personal computer and was designed to use in a Microsoft Excel environment. One of major features available in this software package is the "NeuralWizard" feature. NeuralWizard can be described as a sophisticated neural network builder that sends commands to NeuroSolutions to automatically construct a fully functional neural network. The object-oriented simulation environment of NeuroSolutions gives the user an unprecedented flexibility to construct neural network simulations. The NeuralWizard aids the user by encapsulating the network building rules and reducing the user decisions down to an easy, step-by-step procedure. The following sections summarize the important design and operational hints found in the supplemental material of instructions provided by NeuroDimensions, in combination with our working experience gained from this study.

Using NeuroSolutions for Excel

NeuroSolutions for Excel was designed to allow users to develop a complete solution to their own problem in one simple package. It gives the user flexibility to customize operations using Microsoft's Visual Basic for applications as a scripting language. Figure C1 shows a block diagram describing the order in which the NeuroSolutions for Excel modules can be used to solve the problem. Each step is described below:

(1) Preprocess data module: The main menu item includes differences, randomize rows, sample, moving average, translate symbolic columns, insert column labels.

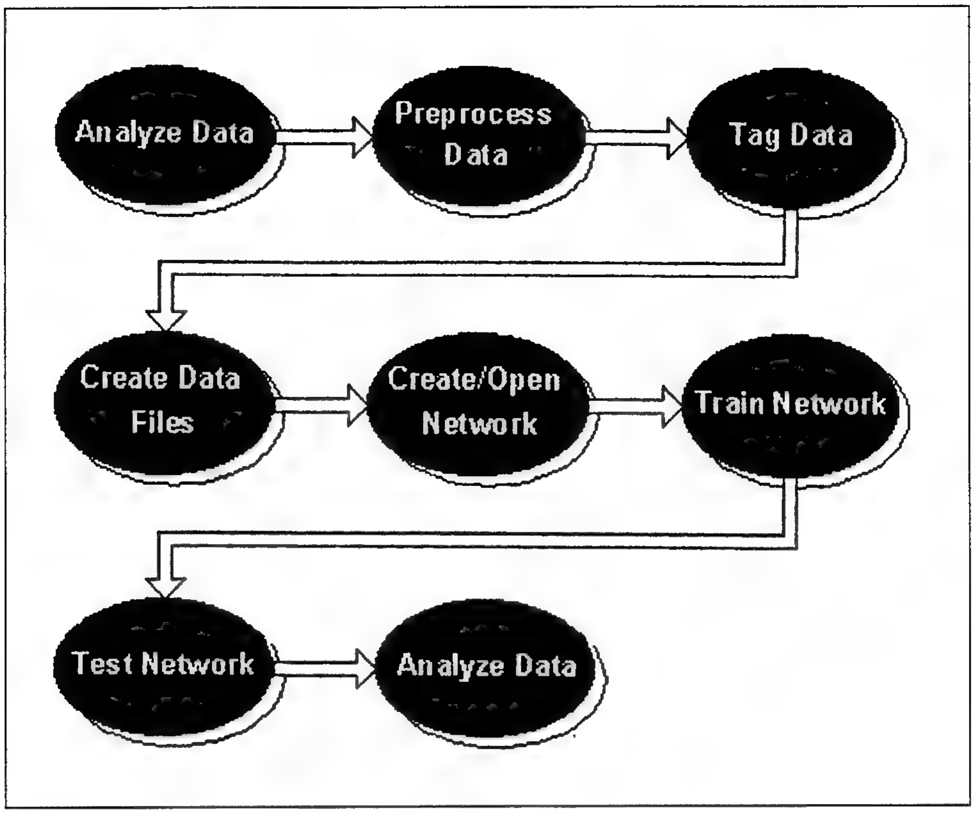


Figure C1. The NeuroSolutions for Excel modules

(2) Analyze data module: It includes the options of correlation, time series plot, xy scatter plot, histogram plot, summary statistics, and trend accuracy. The first three options may be very useful as the screening tool to determine the data patterns and the relationships between the individual input and output functions.

(3) Tag data module: The features of this module are to determine the input/output columns for the ANN model, to select data rows used for training, cross-validation, and testing, and to clear all tags when establishing new testing parameters. Other options allow the user to select a percentage of the data rows to be used for further analysis.

(4) Create data files module: This module is used to create data files either for all tagged cross-sections within the active worksheet or just to create partial data files for a particular purpose, such as training files.

(5) Create/Open network module: Starts the NeuralWizard, which guides the user through the creation of a new NeuroSolutions breadboard. This "breadboard", which is similar in design to that used in building an electrical circuit, serves as a diagram to summarize the configuration of the model structure and flow path. This building process, in turn, will select the different panels within NeuralWizard to select the model neural

network algorithm, learning parameters, non-linear transfer functions, and number of hidden layers and nodes. After a user completes their selection, it can be saved to the active NeuroSolutions breadboard.

(6) Train network module: The active NeuroSolutions breadboard is trained one time and the best network weights are saved. The best network weights are saved at the specified "epoch" (or iteration) when the cross validation error is minimum, if a cross validation data set is used. A report of the training results is then generated. It contains the plot of the training mean-squared error (MSE) versus epochs and the cross validation MSE, and a table showing the minimum training MSE, the epoch at which this minimum training MSE occurred, and the final training MSE.

(7) Test network module: Tests the active NeuroSolutions breadboard on the chosen data set and creates a report of the results. During testing, the learning is turned off and the chosen data set is fed through the network. The contents of this generated report vary based on whether the classification or regression reports type was selected. A table reporting the mean-squared error (MSE), normalized mean-squared error (NMSE), mean absolute error (MAE), maximum absolute error, minimum absolute error, and correlation coefficient for each output is then constructed.

Practical Issues of Learning

The performance of ANN learning is believed to be the most critical issue. There are mainly several practical aspects related to learning. Unfortunately, there are no formulas to select these parameters. Only some general rules apply and a lot of experimentation is necessary.

(1) Training set: The size of the training set is of fundamental importance to the practical usefulness of the network. If the training patterns do not convey all the characteristics of the problem class, the mapping discovered during training only applies to the training set. Thus the performance in the test will be much worse than the training set performance. The only general rules that can be formulated are to use a significant amount of data and use representative data. If it does not have significant amount of data to train the ANN, then the ANN paradigm is probably not the best solution to solve the problem.

Another aspect of proper training is related to the relation between training set size and number of weights in the ANN. If the number of training examples is smaller than the number of weights, it will obviously produce poor generalization. A general rule: The number of training examples is at least double the number of network weights. When there is a big discrepancy between the performance in the training set and test set, it can be suspected as deficient training. Usually, one can always expect a drop in performance from the training set to the test set. In case a large drop in performance (more than 10-15 %), it is recommended increasing the training set size and produce a different mixture of training and test examples.

(2) Network size: Particularly, this refers to choosing the number of input and output nodes within the network, and also specifying the number of hidden layers within that network. At the present stage of knowledge, establishing the size of a network is more efficiently done through experimentation. The issue is the following: The number of processing elements in the hidden layer is associated with the mapping ability of the network. The larger the number, the more powerful the network. However, if one continues to increase the network size, there is a point where the generalization gets worse. This is due to the fact that we may be over-fitting the training set, so when the network works with patterns that it has never seen before the response becomes unpredictable. The problem is to find the smallest number of degrees of freedom that achieves the required performance in the test set.

It is recommended to start with small sized networks and increase the size until the performance in the test set is appropriate. An alternative approach is to start with a larger network, and remove some of the weights. There are a few techniques, such as weight decay, that partially automate this idea. In NeuroSolutions, probing the hidden layer weight activation with the scopes can control the size of the network.

(3) Learning parameters: The control of the learning parameters has been a problem with no solution in ANN research for a long time. The goal is that one wants to train as fast as possible and reach the best performance. Increasing the learning rate parameter will decrease the training time, but will also increase the possibility of divergence, and of routing around the optimal value. Since the weight correction is dependent upon the performance surface characteristics and learning rate, to obtain constant learning, an adaptive learning parameter is necessary. In general, modification of learning rates is possible under circumstances, but several other parameters are included that also need to be experimentally set. NeuroSolutions enables versatile control of the learning rates by implementing adaptive schemes.

The conventional approach is to simply choose the learning rate and a momentum term. The momentum term imposes a memory factor on the adaptation, and has been shown to speedup adaptation while avoiding local minimum trapping to a certain extent.

(4) Stop criteria: Stop learning criteria are based on monitoring the mean square error. The curve of the MSE as a function of time is called the learning curve. The most used criterion is probably to choose the number of iterations, but it can also preset the final error. When, between two consecutive iterations, the error does not drop at least a given amount, training should be terminated. This gives a criterion for comparing very

different topologies. Another possibility is to monitor the MSE for the test set, as in cross-validation. One should stop the learning when error in the test set increases. Usually, this is where the maximum generalization takes place.

To implement this procedure we must train the ANN for a certain number of iterations, freeze the weights and test the performance in the test set. Then, return to the training set and continues learning.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* August 2000	**2. REPORT TYPE** Final Report	**3. DATES COVERED** *(From - To)*
4. TITLE AND SUBTITLE Riverflow/River Stage Prediction for Military Applications Using Artificial Neural Network Modeling		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Bernard B. Hsieh, Charles L. Bartos		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Engineer Research and Development Center Coastal and Hydraulics Laboratory 3909 Halls Ferry Road Vicksburg, MS 39180-6199		**8. PERFORMING ORGANIZATION REPORT NUMBER** ERDC/CHL TR-00-16
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Military Hydrology Program U.S. Army Engineer Research and Development Center 3909 Halls Ferry Road Vicksburg, MS 39180-6199		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Artificial Neural Networks (ANNs) were successfully applied to two different scale watershed systems for riverflow and stage prediction. It is a powerful and easy-to-use operational tool for addressing two of the most difficult temporal and spatial forecasting and prediction problems: nonlinearity and time-delay.

In the lower portions of the Mississippi River, riverflow characteristics at Memphis, TN, can be predicted with a high degree of accuracy from two upstream gauges, even without rainfall data and tributary flow data. Less accurate results were obtained for the Sava River daily flow study, due mainly to the limited length of available data sets. The ANN model performance was excellent for 40 years monthly mean data set for the Sava River. With two upstream sets available, the model can accurately predict the downstream monthly flow. The study indicated that once a good data set is available, it can provide quick and accurate prediction for desired locations, such as the bridge site for military operation.

The best performance of an ANN for flow prediction heavily depends on not only the length of the data sets but also whether the most significant patterns were included in the process.

15. SUBJECT TERMS

Artificial Neural Networks	River stage forecasting	Watershed study
Rainfall-Runoff Model	Riverflow prediction	

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT UNCLASSIFIED	**b. ABSTRACT**	**c. THIS PAGE** UNCLASSIFIED		92	**19b. TELEPHONE NUMBER** *(include area code)*